THESIS

APPLYING MASSIVELY PARALLEL KINETIC MONTE CARLO METHODS TO SIMULATE GRAIN GROWTH AND SINTERING IN POWDERED METALS

by

Aaron M. Hay

September 2011

Thesis Advisor:	Luke N. Brewer
Second Reader:	Young W. Kwon

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2011	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Applying Massively Parallel Kinetic Monte Carlo Methods to Simulate Grain Growth and Sintering in Powdered Metals			**5. FUNDING NUMBERS**
6. AUTHOR(S) Aaron M. Hay			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number: N/A			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Kinetic Monte Carlo (KMC) simulation methods were utilized to study the grain growth and sintering of nanocrystalline metal compacts. Sintering is the process used to fabricate materials from powders by densifying the powder compact at elevated temperatures. Recently, experimental literature has demonstrated that nanoparticles (< 50 nm) can be used to bond materials at dramatically lower temperatures and pressures while maintaining the mechanical properties of nanostructured materials. Despite these promising results, the grain growth and sintering mechanisms of nanostructures are not fully understood.

Simulations performed using KMC algorithms can be used to model nanoparticle grain growth and sintering. Sandia National Laboratories' new, massively-parallel, Stochastic Parallel Particle Kinetic Simulator (SPPARKS) code is capable of simulating large-scale problems of grain growth and sintering from the nanoscale to the microscale.

This thesis focused on setting up SPPARKS on the Naval Postgraduate School's high performance computing resources. The performance of SPPARKS was assessed for large-scale simulations of grain growth and sintering. Using SPPARKS, the ability to perform coupled grain growth and sintering was demonstrated while controlling variables such as temperature, porosity, and grain size. The results demonstrate the importance of the spatial distribution of porosity on the nanostructure evolution during grain growth and sintering.

14. SUBJECT TERMS Grain Growth, Sintering, Nanoparticles, Kinetic Monte Carlo, Stochastic Parallel Particle Kinetic Simulator, High Performance Computer Simulations			**15. NUMBER OF PAGES** 121
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

APPLYING MASSIVELY PARALLEL KINETIC MONTE CARLO METHODS TO SIMULATE GRAIN GROWTH AND SINTERING IN POWDERED METALS

Aaron M. Hay
Lieutenant Commander, United States Navy
B.S., Tulane University, May 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2011

Author:	Aaron M. Hay
Approved by:	Luke N. Brewer Thesis Advisor
	Young W. Kwon Second Reader
	Knox T. Millsaps, PhD Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Kinetic Monte Carlo (KMC) simulation methods were utilized to study the grain growth and sintering of nanocrystalline metal compacts. Sintering is the process used to fabricate materials from powders by densifying the powder compact at elevated temperatures. Recently, experimental literature has demonstrated that nanoparticles (< 50 nm) can be used to bond materials at dramatically lower temperatures and pressures while maintaining the mechanical properties of nanostructured materials. Despite these promising results, the grain growth and sintering mechanisms of nanostructures are not fully understood.

Simulations performed using KMC algorithms can be used to model nanoparticle grain growth and sintering. Sandia National Laboratories' new, massively-parallel, Stochastic Parallel Particle Kinetic Simulator (SPPARKS) code is capable of simulating large-scale problems of grain growth and sintering from the nanoscale to the microscale.

This thesis focused on setting up SPPARKS on the Naval Postgraduate School's high performance computing resources. The performance of SPPARKS was assessed for large-scale simulations of grain growth and sintering. Using SPPARKS, the ability to perform coupled grain growth and sintering was demonstrated while controlling variables such as temperature, porosity, and grain size. The results demonstrate the importance of the spatial distribution of porosity on the nanostructure evolution during grain growth and sintering.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

SGGS – Sandia Grain Growth and Sintering Code

KMC – Kinetic Monte Carlo

MCS – Monte Carlo Step

MD – Molecular Dynamics

N – Number of Sites per Cluster

NEDB – Nanoparticle Enabled Diffusion Bonding

R – Radius

RAM – Random-Access Memory

SPPARKS – Stochastic Parallel Particle Kinetic Simulator

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I am eternally grateful to my wife, Ann, for her encouragement and support during the thesis process. She has been a source of inspiration and dedication. I could not have made it this far without her. I also am thankful of my daughter, Kate, who always brought a smile to my face after a hard day of research and countless hours in front of a computer.

I would like to thank my thesis advisor, Dr. Luke Brewer, for allowing me the freedom to find my way through the thesis process. His expertise in the field of materials and computer simulations was instrumental in completing this thesis. I would also like to thank Dr. Young Kwon for offering his time and efforts in reviewing my theses.

The following work would not have been possible without the technical expertise of Dr. Veena Tikare, Dr. Steve Plimpton, and Christina Garcia Cardona at Sandia National Laboratories.

I would also like to thank Milan Vukcevich and Brian Andrus for teaching me the skills necessary to navigate the world of high performance computing. Without their assistance, I would still be trying to learn how to type commands in Linux.

I also want to thank Dr. Fittante for giving me the tools necessary to excel in my studies and keep my composure when my research did not go as planned.

Finally, I would like to thank all of my professors at Naval Postgraduate School for sharing their time and expertise during my classroom studies. I especially thank Dr. Sanjeev Sathe, Dr. Sarath Menon, LtCol Randall Pollack (USAF), CDR Jon Vanslyke, and CDR Dan Chisholm.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The ability to fabricate structural components from metals with a fine (micron-sized), controlled grain size is one of the hallmarks of modern, structural metallurgy. Powder metallurgy, in particular, consists of powder manufacture, powder blending, compacting, and sintering. There is a wide range of applications for powder metallurgy. Powder metallurgy is important in the manufacture of heat shields for spacecraft reentry into the Earth's atmosphere, linings for friction brakes, turbine disks, and metallic glasses for high-strength films and ribbons to name a few. All of these applications are enabled by materials with high yield strengths and toughness, both of which are increased with a reduction in grain size. Recently, there has been an exploration of the reduction of grain size in metals down to the nano-scale. One of the primary methods for fabricating macroscale components from nanoscale powders is by the sintering of nanosized particles [1]. Even though the sintering of nanoparticles is similar to the sintering of coarser particles, there are unique issues and challenges involved with sintering nanosized particles. One major challenge in sintering of nanoparticles is to achieve the maximum density of the materials while retaining the properties of the nanoparticles [1]. Therefore, the goal of sintering nanoparticles is to minimize grain growth while densifying the material. Unfortunately, many of the factors which result in densification also lead to grain growth and the possible loss of the desired properties of the nanomaterial. The grain growth experienced during heating of the particles to the sintering temperature may be enough to change the properties of the nanomaterials and thus render the process unsuccessful. Preventing grain growth while generating densification is critical to successful powder metallurgy at the nanoscale.

A recent and related process is nanoparticle-enabled diffusion bonding (NEDB). The use of nanoparticle metals for diffusion bonding allows for joining at very low temperatures and stresses. Ide et al. joined two copper discs using silver metallo-organic nanoparticles [2]. The 11 nm silver nanoparticles produced an interfacial bond microstructure between the copper discs that was stronger and denser than that using fine silver particles of 100 nm [2]. The joints made with the silver nanoparticles had shear strengths of 25–40 MPa with fractures located inside the sintered silver layer. Conversely, the joints made with the fine silver particles had shear strengths of only 0.5–10 MPa and fractured at the copper-to-silver interface. Of particular importance were the surprisingly low pressure and temperature required to form the sintered bonds of the silver nanoparticles, 1 and 5 MPa and 573 K, respectively [2]. These pressures and temperatures are well below the values required for diffusion bonding with larger particles [3]. A number of recent papers have extended these results towards use in bonding and packing electronic devices using silver at low temperatures and pressures [4], [5], [6].

A final example of the importance of nanocrystalline metals is the production of metallic materials with extremely high yield stresses. According to the Hall-Petch equation, the strength of a material goes up considerably when the size of the particles is reduced to the nanoscale. The Hall-Petch equation has the form:

$$\sigma_y = \sigma_o + kD^{-1/2} \tag{1}$$

where σ_y is the yield stress,

σ_o is the frictional (Peierls) stress required to move dislocations,

k is the Hall-Petch slope, and

D is the grain size [7].

The yield stress is proportional to the inverse of the square root of the grain size. For silver, the yield strength increases from 47.26 MPa for 50 μm-sized particles to 350.41 MPa for 50 nm-sized particles:

$$\sigma_y = 37.36MPa + \frac{0.07MN/m^{3/2}}{\left(50x10^{-6}m\right)^{1/2}} = 47.26MPa \tag{2}$$

$$\sigma_y = 37.36MPa + \frac{0.07MN/m^{3/2}}{\left(50x10^{-9}m\right)^{1/2}} = 350.41MPa \tag{3}$$

This is a 741% increase in the yield strength of the material. Using this calculation of the yield strengths of a material reveals the importance of being able to sinter at the nanoscale. Myers and Chawla have described the high strengths associated with a reduction of grains to the nano-scale. They revealed that a strength level of a drawn steel with a grain size of 10 nm was 4,000 MPa, approximately four times higher than for a typical high strength steel. Several reasons for the high strengths have been formulated by Meyers and Chawla such as dislocations pileups, dislocation network models, and grain-boundary sliding [7]. It has been shown that the Hall-Petch slope decreases as the grain size is reduced. The yield stress still goes up considerably as the grain size goes from 100 nm to 10 nm as has been shown for copper in the work of Weertman et al. (Figure 1) [8]. Albert et al. were able to achieve tensile strengths in silver nanoparticle samples greater than 100 MPa. These samples were sintered at lower temperatures, 125–175 °C, using a compressive forces of 600 N [9]. These examples show the possibilities of sintering nano-scaled powders.

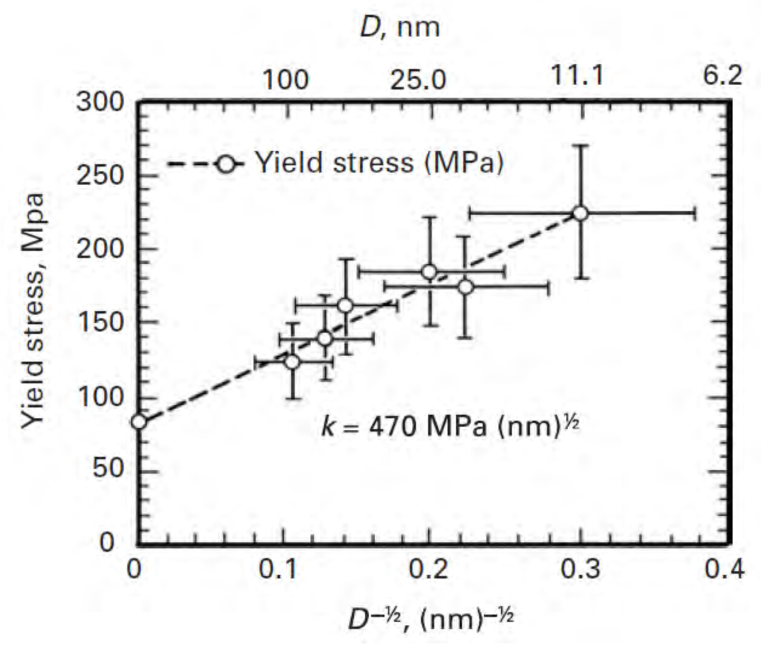


Figure 1. Hall-Petch Relationship for Nanocrystalline Copper From [7].

B. CHALLENGES OF BONDING NANOPARTICLES

Despite the promise of nanocrystalline materials, synthesis of bulk materials from nanoparticles is difficult for several reasons. The key challenge is to form a bulk material (dimension in centimeters to meters) from powder particles that are less than 100 nm, i.e., 10^7 times smaller. Sintering, hot pressing, extrusion, and even rolling can be used to consolidate the powders and form them into a useful shape; however, these manufacturing processes lead to potentially serious difficulties [1]. The first challenge presented with fabricating materials at the nanoscale is retaining the nanocrystalline structure itself. As discussed by Fang and Wang, nanocrystals can easily experience such rapid grain growth during the sintering process that the nanoscale grain size cannot be maintained [1]. This uncontrolled grain growth is perhaps the greatest challenge to sintering at the nanoscale level.

The next concern is preventing the melting of the nanoparticles. Studies have shown that the melting temperature of very fine nanoparticles decreases with the size of the particle [10], [11], [12], [13], [14]. These papers have revealed depressed melting temperatures as much as 50% of the bulk material for very small (< 20 nm) particles. Failing to account for the depressed melting point when using conventional sintering temperature ranges could melt the nanoparticles, resulting in the loss of the nanocrystalline structure and the resulting properties. The depressed melting temperature of nanosized metals is not fully understood, but is likely related to the higher surface energy to volume ratio. Even with this ratio known, it is difficult to predict the melting temperature and therefore, the temperature required for sintering at the nanoscale.

Another challenge in producing component materials with nanoparticles is oxidation. Many metals readily oxidize at low temperatures, and metallic nanoparticles do so even more readily due to their extremely high surface area to volume ratio. Aluminum is a good example of a material that oxidizes extremely rapidly. Extra care must be taken to prevent the aluminum nanoparticles from oxidizing. If the metallic nanoparticles oxidize prior to consolidation, then the oxide layer will interfere with or even prevent sintering between particles. Several approaches include sintering in an atmosphere that does not contain oxygen (e.g., using an inert gas or vacuum) [9], [15], [16], [17], [18]. Another approach is to coat the nanoparticles with a compound that resists oxidation; although, coating the nanoparticles can introduce the additional problem of developing and removing the coating so it does not interfere with the sintering process [2].

C. SINTERING FOR CONSOLIDATING NANOPARTICLES

Several solutions exist for successfully creating macroscale materials out of consolidated, nanoparticle powders. As mentioned previously, one solution is sintering which is the surface diffusion-driven densification of powders at elevated temperatures. This thesis will focus mainly on the formation of component materials from nanoscale powders using sintering. Sintering involves the heating of a material to produce bonding between powder particles, which reduces the overall energy of the material. The bonding that occurs during sintering strengthens the material and produces improved engineering properties of the compacted particles [19]. Sintering is used routinely in the fabrication of both metals and ceramics from powders and is, in fact, the dominant processing route for ceramic materials [19]. As sintering is driven by the reduction of surface area, nanoparticles readily sinter. The sintering of nanoparticles has been successfully demonstrated at relatively low temperatures and pressures while maintaining the mechanical properties of the nanoparticles [1]. However, the challenges mentioned above (rapid grain growth, particle size-dependent melting point, oxidation of particle surfaces, etc.) can limit the densification and nanostructure retention in sintered nano-powder compacts. While sintering using micron-scale particles is well understood, there is still relatively little known about the physics and mechanisms of using nanoparticles to form larger materials for useful applications. This thesis is centered on the theory of sintering at the nanoscale and using computer modeling to predict the mechanisms of nanoparticles sintering.

There are many questions concerning the sintering at the nanoscale. The most fundamental is whether the sintering of nanoparticles is different from conventional sintering on the microscale. The mechanism of nanoparticles sintering and diffusion bonding may well be different than at larger length scales. The sintering densification rate (dV_s/dt) is a key parameter to distinguish between conventional sintering and sintering at the nanoscale. The densification rate in the intermediate stage of sintering is given by:

$$\frac{dV_s}{dt} = \frac{g\gamma_{SV}\Omega D_V}{kTG^3} \tag{4}$$

where V_s is the fractional volume of the solid,

t is time,

g is a collection of geometricalgeometric terms with a value typically near 5,

γ_{SV} is the solid-vapor surface energy,

Ω is the atomic volume,

D_V is the volume diffusion coefficient,

k is Boltzmann's constant,

T is the absolute temperature,

and G is the grain size [19].

According to the above equation, the densification rate is inversely proportional to the cube of the grain size and the inverse of temperature. Albert et al. predicted the effect of grain size and temperature on the densification rate (Figure 2). Smaller grain sizes and higher temperatures result in faster densification rates. Albert et al. used a sintering pressure of 500 MPa and started at a density of 90%. From Figure 2, at 100°C, decreasing the grain size from 500 nm to 5 nm increased the densification rate seven orders of magnitude. The densification rate is also predicted to increase five orders of magnitude by raising the temperature from room temperature to 175°C [9]. Even though these results are pressure-assisted, the results show the increase in densification rate at the nano-scale.

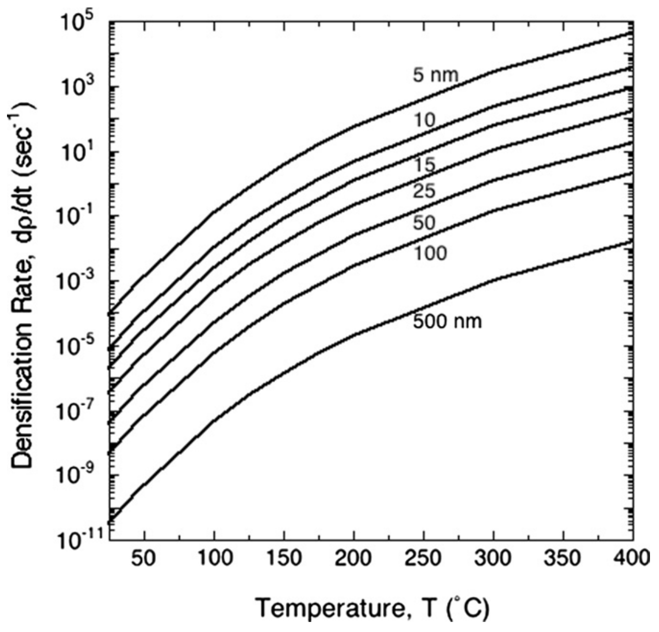


Figure 2. Densification Rate as a Function of Grain Size and Temperature From [9].

Fang and Wang described that the densification behavior of nanoparticles during sintering is different than conventional sintering with respect to the densification rate and the temperature range of the densification. They stated that agglomeration, pores, and other variables affect sintering at all length scales, however, the effects are more pronounced with nanoparticles [1]. They showed that the vacancy concentration is non-linear when particles reach the nano-scale:

$$\Delta C_V = C_{V0}\left[\exp\left(-\frac{\gamma\kappa\Omega}{kT}\right)-1\right] \tag{5}$$

where ΔC_V is the change in vacancy concentration,

C_{V0} is the initial vacancy concentration,

γ is the surface energy of the material,

κ is the curvature of the surface,

Ω is the atomic volume,

k is Boltzmann's constant, and

T is the absolute temperature [1].

Figure 3 shows the vacancy concentration as a function of particle size. As the particle size decreases, the vacancy concentration deviates from the linear model. The horizontal axis is $1/d^*$, which is related to the particle size and equal to $-\frac{\gamma\kappa\Omega}{kT}$. The diffusivity term, D_v, is increased as the vacancy concentration increases. Therefore, the densification rate increases as the particle size gets smaller.

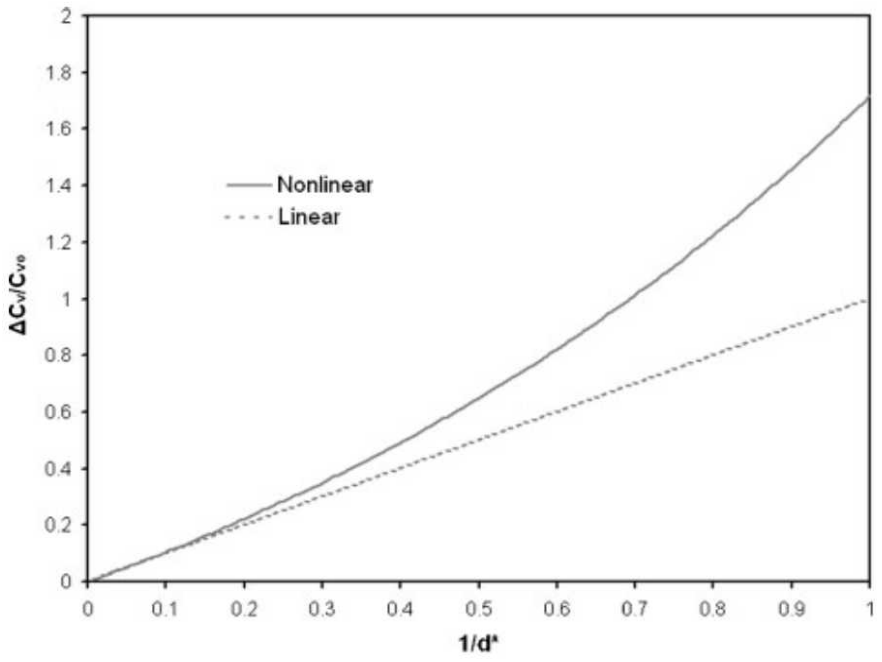


Figure 3. Vacancy Concentration as a Function of Grain Size From [1].

In addition to the grain size, temperature is an important parameter for sintering. The sintering temperature is the range of temperatures when bonding occurs between the particles. The homologous temperature is the sintering temperature of a material normalized by the absolute melting temperature. According to German, the homologous temperature during sintering is between 0.5 and 0.8 for most materials [19]. From the literature above, the homologous temperature for silver nanoparticles ranges from 0.3–0.5 based on a melting temperature of 1,234K. As stated previously, the melting temperature of nanoparticles is depressed from the melting temperature of the bulk material. Therefore, the lower melting point of nanoparticles results in a different temperature range for sintering with nanoparticles vice with microparticles.

Both grain growth and sintering are important when bonding metal powders. Fang et al. described that the key to sintering nanoparticles as the ability to limit grain growth while encouraging sintering. Grain growth is important and can dominate the processing of nanomaterials [1]. Therefore, understanding the competition between grain growth and sintering is vital to ensuring the proper sintering of nanomaterials.

D. IMPORTANCE OF COMPUTER SIMULATIONS

Computer simulations of sintering allow the examination of the fundamental processes of grain growth and sintering and their interplay. Using computer simulations is important for decoupling the individual, physical mechanisms from each other. An example of the difficulties of sintering nanoparticles in physical experiments is the growth of the grains during the heating process from room temperature to the sintering temperature [1]. Grain sizes of 10 nm could grow considerably by the time sintering actually begins, thus complicating the ability to understand the actual sintering mechanisms at the 10 nm particle scale.

Decoupling the processes of grain growth and sintering is relatively more straightforward using computer simulations. This thesis will show how simple it is to isolate the process of grain growth or sintering. Other parameters can be varied utilizing computer simulations without affecting other parameters. For instance, the grain growth and sintering temperature can be set without changing the rates for pore migration or annihilation, which are dependent on the temperature in physical experiments. Conversely, the probability of pore migration and annihilation can be determined on the sintering of nanoparticles without changing the temperature. It is clear from these examples that using computer simulations is important to understanding the individual parameters on grain growth and sintering and their overall effect on the system.

Another important reason for using computer simulations is the ability to access physical and microstructural variables which are extremely difficult to control or measure experimentally. Several important parameters with grain growth and sintering are the grain size and neck size. As described in the next section, the grain size is important in determining the rate of grain growth and the neck size controls the rate of sintering and densification of the system. Both parameters are difficult to access during experiments while sintering is occurring, but computer simulations allow these parameters to be recorded throughout the simulation, in three dimensions, and at any length scale. The density can also be recorded throughout the simulation, which allows for the densification rate of the particles during sintering. Determining the density during sintering experiments is difficult and measuring the spatial distribution of the porosity is exceptionally difficult, particularly at the nanoscale.

Simulations can also examine the effects of various physical parameters on the same initial nano- or microstructure. For instance, a simulation can be run with grain growth and sintering and stopped at a predetermined time (e.g., particular density or grain size). The simulation can then be restarted from the end of the previous simulation and parameters such as temperature changed to determine the effects of changing one parameter on the rest of the simulation. In addition, simulations can be extended beyond the end time if more data is required. By saving the final data, a new simulation can be started without the need to start from the beginning. The ability to begin numerous simulations with identical starting microstructures is also extremely useful. All of these abilities are extremely difficult to accomplish in actual experiments and often impossible. For example, starting numerous experiments from identical microstructures is impossible in laboratory experiments. Finally, the ability to run simulations in parallel allows for multiple results by changing a few parameters at a time.

E. IMPORTANCE OF KINETIC MONTE CARLO

There are several methods available to simulate grain growth and sintering of particles. One method is molecular dynamics. Molecular dynamics (MD) simulations generate information at the atomic scale. This includes a comprehensive tracking of all atomic positions and velocities from which detailed defect structures and system thermodynamics can be calculated [20]. However, MD by itself is not sufficient to study sintering because the time scale is too short (e.g., 50 nanoseconds is a very long MD simulation, while sintering proceeds in minutes to hours). MD has been successfully used to determine diffusivities that can be incorporated into meso-scale simulation techniques such as kinetic Monte Carlo (KMC) [21]. The finite element method is useful for solving problems in solid mechanics, heat transport, and other fields. However, FEM is primarily a continuum technique and is not well-posed, by itself, to simulate the microstructural mass transfer and evolution inherent in the sintering process [21]. Therefore, a method situated between the scope of molecular dynamics and finite element methods is required.

Kinetic Monte Carlo is an ideally suited simulation technique for modeling grain growth and sintering. Grain growth and sintering simulations require many atoms or particles, long simulation times, and high temperatures. If a simulation needs 1000 particles to adequately model sintering, then there needs to be 10 particles per side of a cubic simulation volume. In order to adequately model the physics of a particle at the quasi-continuum scale, approximately 10 voxels (three dimensional pixels) are required across each grain. This results in a cube that has 100 sites per edge, as shown in Figure 4, which is represented by a simulation volume with 10^6 voxels, a very manageable size simulation for even a good lap-top computer. A voxel is defined as the basic three-dimensional cube in the simulation volume and is analogous to the two-dimensional pixel. A particle needs to have a diameter of at least 10 voxels to accurately model the grain growth and sintering of the grains. If the side of the model volume is set at 500 nm, then each voxel is 5 nm across and each particle has a diameter of 50 nm.

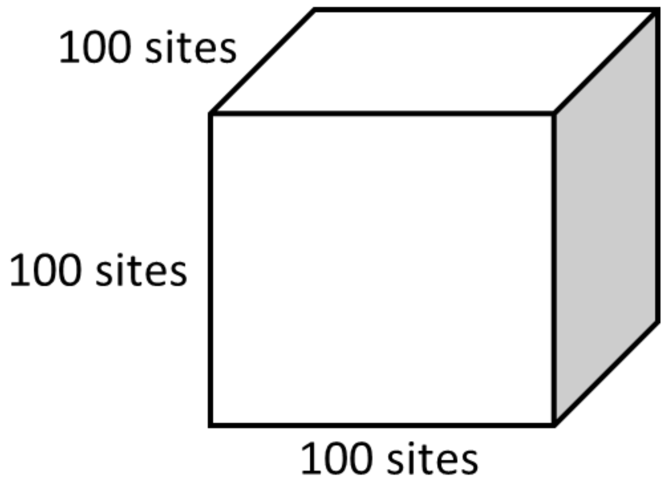


Figure 4. Computer Simulation Volume Representation.

Based on experimental data, nanoparticles lose their nanoscale properties at approximately 100 nm in average diameter [1]. In order to capture the transition between nanoscale effects and microscale sintering and grain growth processes it is essential to start with nano-size crystals (< 50 nm) and have them coarsen to a micron-sized particle (> 100 nm) during the simulation. Modeling this transition results in a large increase in the number of sites required per side. For the example in Figure 4, the minimum number of 50 nm particles would be 10 per side if the side is set at 500 nm. This results in 1,000 grains in the volume. Growing the grains to 100 nm would result in five grains per side for a total of 125 grains. Increasing the volume allows for smaller grains and the ability to sufficiently grow the grains to micron-sized particles. If each side had 500 sites, then a voxel would be 1 nm on a side for a 500 nm sized simulation. Having a diameter of 10 voxels, 10 nm particles could be simulated. This would result in 50 particles per side for a total of 125,000 particles. Sintering to a particle size of 100 nm would result in 125 grains. Therefore, increasing the number of sites per side from 100 to 500 allows for 10 nm grains to be coarsened to 100 nm vice the larger 50 nm starting grain size. To give an idea of the simulation sizes, silver is used as the sintering metal. For silver, the number of atoms per 1 nm^3 is given by:

$$\frac{atoms}{nm^3} = \left(\frac{6.023x10^{23}\,atoms}{mol}\right)\left(\frac{10.49g}{cm^3}\right)\left(\frac{mol}{107.87g}\right)\left(\frac{cm}{1x10^7\,nm}\right)^3 = 58.573\frac{atoms}{nm^3} \quad (6)$$

This results in 7.32 x 10^9 silver atoms per 500^3 site simulation box. A 500^3 site simulation would only contain 1.31 x 10–^{12} grams of silver. In order to simulate a single gram of silver, over 4.5 million sites per edge would be required.

The simulation sizes required to model sintering presents a unique challenge for computer simulations. The ability to rapidly and accurately model grain growth and sintering is vital to generating useful results. KMC algorithms have traditionally been run on serial codes, which choose sites at random one after the other. This serial approach to KMC slows down the process and limits the size problems that can be run. In order to speed up the process and run larger simulations, Sandia National Labs developed the ability to implement KMC in parallel. Massively parallel KMC solvers enable larger problems to be simulated accurately and efficiently due to the large number of processors that can be used. Sandia developed on open-source code known as the Stochastic Parallel Particle Kinetic Simulator (SPPARKS), which can partition the simulation across numerous processors, communicate information between processors, and output snapshots of the simulations at predetermined times [22]. The Sandia National Laboratories grain growth and sintering (SGGS) code was written by Dr. Veena Tikare and Christina Garcia Cardona as an add-on to SPPARKS.

F. THESIS OBJECTIVES

The work in this thesis assesses the potential of the SPPARKS KMC code to simulate grain growth and sintering. The objectives of this thesis are as follows:

1. Learn and set-up the SPPARKS code at Naval Postgraduate School (NPS).

The SPPARKS code requires the identification of high performance computing resources, compilation and queuing on those resources, and means for handling the large input and output data all on the resources available at NPS.

2. Assess the performance of the SPPARKS code at NPS for large-scale simulation of grain growth and sintering of nano-scale materials.

This objective will quantitatively assess the computational performance of the SPPARKS code on large materials simulations using NPS resources identifying limits of simulation size, time, and sensitivity of results to simulation size.

3. Identify and test key physical variables required to properly model grain growth and sintering using kinetic Monte Carlo codes such as SPPARKS.

While SPPARKS has been used successfully for basic grain growth modeling and for sintering simulations, we need to identify which physical parameters are currently included in the model and which are not but should be in the future. An assessment of the handling and accuracy of these parameters will also be addressed.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. SINTERING THEORY

1. Overview

Sintering is the process of consolidating powder particles into a dense solid by activating surface diffusion at high temperatures. Powders are used because they have fluid-like characteristics which allows shaping or molding under a wide range of stresses [19]. Sintering can be conducted on numerous materials including ceramics, metals, and plastics. The bonds formed during sintering reduce the surface energy of the particles by reducing the free surface areas of pores between particles. As the heating process is continued the pore volume reduces and the density goes up. Increased density and reduced pore volume is desirable during sintering. Depending on the application, shrinkage caused by these processes may be not be desirable [19].

Sintering has many advantages for producing materials from powders with desirable properties. As described above, the chief advantage of sintered materials is that the grain size is set by the powder size and generally much smaller than can be achieved by mechanical metallurgy methods. Another advantage of sintering includes the high purity and uniformity of the initial material, and the preservation of this purity throughout the process. A final advantage of sintering is the ability to reproduce the results due to the control of the initial grain size and sintering conditions, such as pressure and temperature.

One aspect of sintering that is important is the distinction between densification and coarsening. Densification is a critical aspect of sintering. In order to achieve improved properties, the sintered particles will likely need to be densified from the green density of about 65–75% to a final density over 90%. Sintered materials result in a reduction of surface area, grain size increase, and compact strengthening through a process known as coarsening [19]. Coarsening is characterized by the grains growing larger, and smaller pores coalescing to form larger pores. Many materials exhibit both densification and coarsening throughout the sintering process. Coarsening is favored over densification when the grain size is small and the pores are large with a high coordination number. Developing the ability to densify the material, coarsen the compact, or a combination of the two is important to developing a final product with the desired engineering properties.

The chief challenge in sintering materials for all powder sizes is complete densification. Failure to achieve a fully-dense material can seriously degrade the mechanical properties of the component material. Initially, a loose compact of particles has a green density of about 60–74% with the maximum given by:

$$\text{Green Density Maximum} = \frac{Volume_{sphere}}{Volume_{cube}} = \frac{4\left(\frac{4}{3}\pi r^3\right)}{16\sqrt{2}r^3} = 0.7405 \quad . \tag{7}$$

For many applications the final desired density is between 90–95%. In order to achieve the desired densification, the driving force for mass transport must be maximized. The key driving force for mass transport during sintering is the sintering stress.

2. Sintering Stress

The sintering stress is important in describing the rate of sintering at each stage. The stress associated with sintering is due to the curved surfaces of the individual particles. The sintering stress is given by the Laplace equation as:

$$\sigma = \gamma\left(\frac{1}{R_1} + \frac{1}{R_2}\right) \tag{8}$$

where γ is the surface energy, and

R_1 and R_2 are the principal radii of curvature for the surface (Figure 5).

Figure 5 shows the principle radii of a curved surface at a general point. According to German, the stress depicted in Figure 5 is tensile because the radii are located inside the mass (surface curvature is convex). For radii outside the mass, the stress is compressive and the sign is negative (surface curvature is concave). A flat surface has infinite radius and is stress free [19]. The driving force for sintering is to flatten the surface in order to reduce the sintering stress. Therefore, surfaces with bumps or dips will flatten over time as atoms are removed or deposited at the surface.

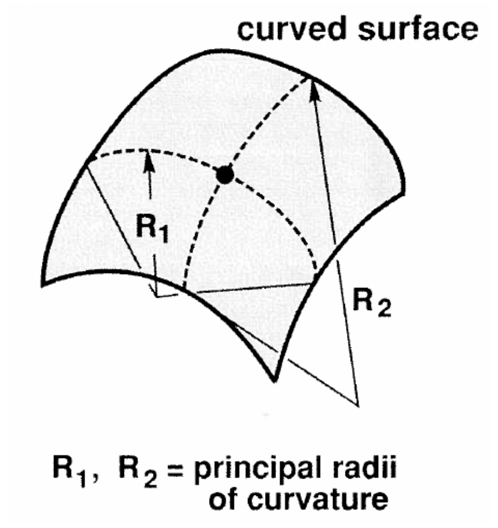


Figure 5. Principle Radii of a Curved Surface From [19].

From Equation 8, the sintering stress is highest for small particles and decreases as the particle size gets larger. Figure 6 shows the sintering stress as a function of the grain size. The sintering stress increases four times when the grain size is reduced from 20 nm to 5 nm. Therefore, the driving force for very small grains (< 20 nm) is much larger than for micron-sized grains (> 100 nm).

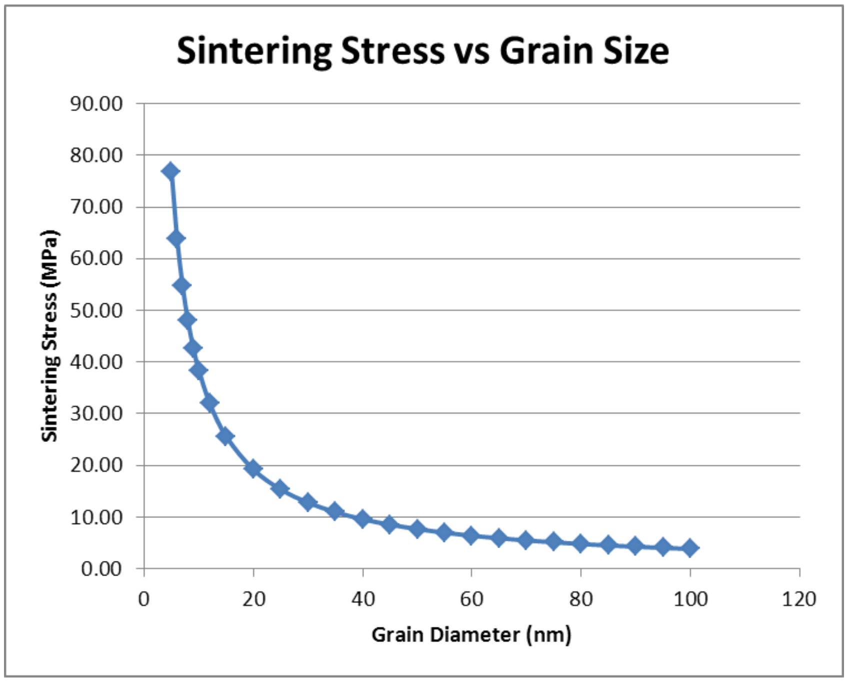


Figure 6. Sintering Stress as a Function of Grain Size.

Figure 7 shows the sintering of two spherical particles of equal size and diameter, D. The neck is defined as the circle of diameter, X, formed by the bonding of the two spherical particles. As an example, the neck growth between two spherical particles can be quite rapid early in the sintering process. German states that the stress given by two spherical particles of equal radii during sintering is:

$$\sigma = \frac{4\gamma}{D}. \tag{9}$$

If radius of the neck is approximated as $X^2/4D$, then the curvature of the neck results in the following sintering stress:

$$\sigma = \gamma\left(\frac{1}{X} - \frac{4D}{X^2}\right) \tag{10}$$

where X is the neck diameter [19]. The smaller neck diameter results in extremely large sintering stresses and the driving force for sintering is high. As sintering progresses, the neck size becomes larger and the sintering stress goes down. The increase in grain size will also lower the sintering stress and reduce the driving force for sintering.

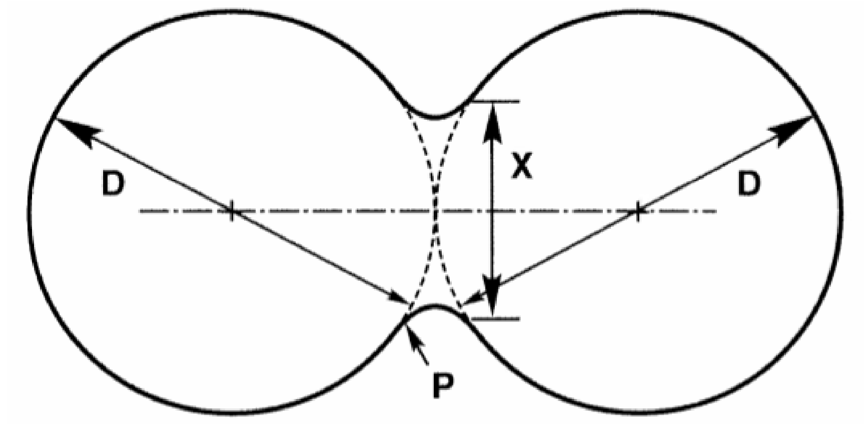


Figure 7. Sintering Profile for Two Spherical Particles From [19].

From Equation 10, the stress gradient in the neck is large due to the sign change in the radius of the circles over a short distance [19]. The pores can have concave or convex curvature (indicated by P in Figure 7). The second term in the above equation is positive if the pore is convex and negative if the pore is concave. From Equation 10, it is evident that the first term is dominant at small grain sizes and large pores, while the second term is dominant at larger grain sizes and small pores. For pressure-assisted sintering, the pressure term becomes dominant as the pressure acts to close pores and densify the mixture while minimizing grain growth [19]. There are many factors affecting the sintering stress, and the stress depends on these factors differently as the sintering progresses through the stages.

3. Stages of Sintering

There are four main stages of sintering. Figure 8 shows the progression of sintering from a loose powder, to the neck growth during the initial stage, to the densification in the intermediate stage, and finally to the decrease in pore size and grain growth of the final stage. Table 1 lists the sintering stages and the key parameters associated with each stage. Figure 9 shows the pore evolution of the pores during sintering. The pores begin as a series of connected spaces between the particles and eventually become closed off as the particles densify. The convention for describing the pores is initially the pores are open and become closed as the material densifies and the pores become isolated from each other.

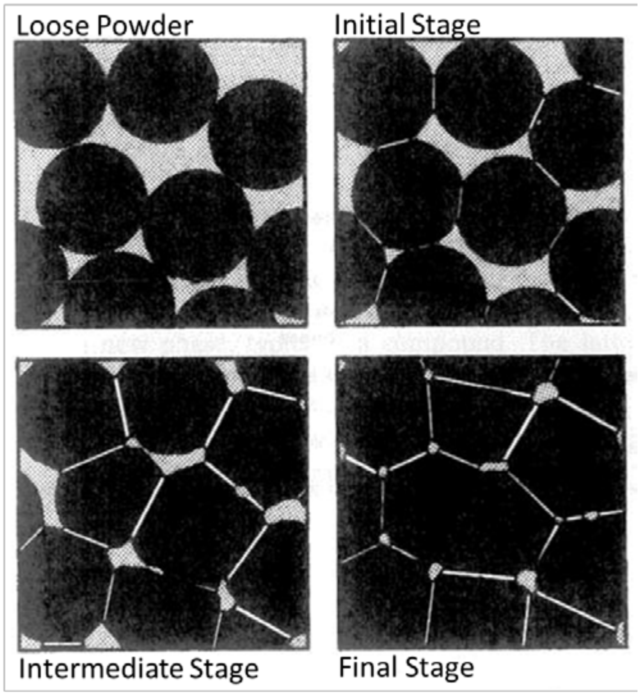


Figure 8. Stages of Sintering From [19].

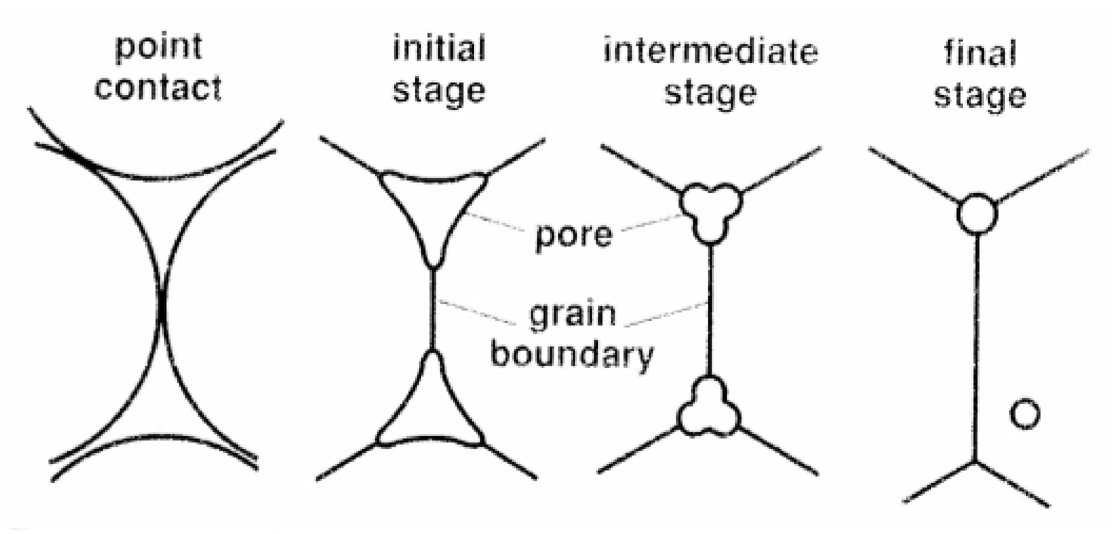


Figure 9. Pore Structure Evolution During Sintering From [19].

The first stage of sintering is the adhesion, rearrangement, and repacking of particles. Particle adhesion occurs spontaneously as the particles are prepared for sintering. The particles then rotate and repack to obtain a higher green density and to lower the energy of the grain boundary structure. The grains rotate and twist in response to unbalanced surface energies, inhomogeneous packing coordination, and different crystallographic orientations [19]. The initial coordination number of each grain is approximately seven with a green density of approximately 64%, and the coordination number increases to a maximum of 14 as shrinkage and densification occur.

Stage	**Process**	**Surface Area Loss**	**Densification**	**Coarsening**
Adhesion	Contact formation	Minimal unless compacted at high pressure	None	None
Initial	Neck Growth	Significant (up to 50% loss)	Small at first	Minimal
Intermediate	Pore rounding and elongation	Near total loss of open porosity	Significant	Increase in grain size and pore size
Final	Pore closure, final densification	Negligible further loss	Slow and relatively minimal	Extensive grain and pore growth

Table 1. Stages of Sintering From [19].

The second stage of sintering is the initial stage or neck growth. During this stage, the necks formed between the particles grow as the particles are heated. The temperature required to conduct sintering is between 50% and 80% of the material's melting temperature, which can be quite high for some materials such as metals. Because the necks are extremely small, they grow independent of each other during the initial stage. The neck growth stage ends when the necks begin to interact and the ratio of neck diameter to particle diameter is approximately 0.3. Neck growth is important because it can be linked to other parameters such as shrinkage, surface area, and density [19].

The intermediate stage is the most important stage for densification. It is characterized by the simultaneous pore rounding, densification, and particle grain growth [19]. The densification rate was given by Equation 4 and shows that smaller grains aid in the densification by increasing the rate as $1/G^3$. There is rapid grain growth at the end of the intermediate stage due to the diminishing pore pinning effect as the pores shrink from a cylindrical shape to a spherical shape and occupy less of the grain boundary area [19].

The final stage is characterized by increasing density as the pores close and the open pore percentage goes down. The open pores begin to close at approximately 15% porosity and are completely closed by 5%. The final stage also exhibits Ostwald ripening, where larger pores grow at the expense of smaller pores [19]. The final stage is a slow process compared to the first three stages. The final stage sintering stress (σ) is a function of both the grains and the pores and is given by the following equation:

$$\sigma = \frac{2\gamma_{SS}}{G} + \frac{4\gamma_{SV}}{d_p} \tag{11}$$

where G is the grain size

γ_{SS} is the solid-solid grain boundary energy,

γ_{SV} is the solid-vapor surface energy, and

d_p is the pore size [19].

The rate of grain growth is given by:

$$\frac{dG}{dt} = \frac{4K_f \gamma_{SV} M_P}{G\left(M_P / M_G + 1\right)} \tag{12}$$

where K_f is a geometric constant that relates the pore spacing and the grain boundary curvature and is typically near unity,

M_P is pore mobility, and

M_G is grain boundary mobility [19].

4. Mass Transport Mechanism

The mass transport mechanisms describe the movement of mass in response to the driving force of sintering. There are two classes of mass transport: surface transport and bulk transport. The two classes are composed of several mass transport mechanisms at the atomic level (Table 1). Conventional sintering involves the motion of vacancies for describing the phenomenon of pore elimination. Vacancies and atoms can move by surface diffusion, evaporation-condensation, grain boundary diffusion, viscous flow, or volume diffusion. The key difference between surface and bulk transport is the densification or shrinkage of the particles, which does not occur during surface transport mechanisms [19].

Category	**Mechanisms Involved**
Surface Transport	Evaporation-Condensation Surface Diffusion Volume Diffusion
Bulk Transport	Plastic Flow Grain Boundary Diffusion Volume Diffusion

Table 2. Mass Transport Mechanisms From [19].

Neck growth occurs during the surface transport processes due to the movement of atoms from the surface of the particles to the neck surface. Because the atoms come from the surface of the particles and not the interior, there is no densification or shrinkage. The dominant mechanisms for surface transport are surface diffusion and evaporation-condensation. Surface diffusion dominates during sintering of most metals at low temperatures [19]. Evaporation-condensation is not as prominent and does not occur in metals. Volume diffusion can occur but the transport of mass is extremely slow compared to the other mechanisms because the atoms must move through the interior of the particle vice the surface of the particle [19].

Bulk transport mechanisms also contribute to neck growth, but result in the densification of the particles due to the atoms originating at the interior of the particles and depositing at the neck. Volume diffusion, grain boundary diffusion, plastic flow, and viscous flow all contribute to bulk transport. Plastic flow usually occurs early in the heating process while grain boundary diffusion is important to the densification of crystalline materials. As a general rule, the bulk transport mechanisms are more active at higher temperatures due to the movement of atoms within the interior of the particles vice on the surfaces. Only those mass transport phenomena associated with the sintering of metals will be discussed.

a. Surface Diffusion

The first diffusion mechanism is surface diffusion. Surface diffusion is dependent on temperature and crystal orientation. Typically, the surface of a particle is not smooth and contains elements such as kinks, ledges and vacancies, where atoms can be easily removed or deposited. There are three steps for surface diffusion: 1) breaking of atomic bonds, 2) random atomic motion, and 3) the reattachment of the atoms to a new surface. The movement of the atoms across the surface of the particles is extremely fast and the limiting mechanism is the breaking or making new atomic bonds. The surface diffusion energy determines the rate of the slowest step and is a function of temperature. Surface diffusion increases as the temperature is increased and grain size is reduced because the amount of surface defects increases which leads to more surface diffusion [19]. Since the surface diffusion activation energy is typically lower than the other activation energies, surface diffusion occurs at lower temperatures and is the most prominent mechanism while the sintering temperature is reached.

During surface diffusion, the atoms move from one surface site to another site. Therefore, surface diffusion does not result in densification. This makes surface diffusion undesirable in processes where densification is important, but highly desirable when shrinkage is unfavorable. When densification is desired, rapid heating to the sintering temperature is one method of minimizing the mass transport due to surface diffusion. As sintering progresses through the stages, surface diffusion becomes less of a factor in mass transport. It still plays a role in pore migration and can act in concert with other mass transport mechanisms later in the sintering process.

b. Volume Diffusion

Volume diffusion is the movement of vacancies through the lattice structure of a particle. The rate of volume diffusion is dependent on temperature, composition of the particles, and curvature or pressure for pressure-assisted sintering. In metals, the temperature is the key factor in determining the diffusion rate; however, as noted by Fang, the diffusion rate can also be increased in nanocrystalline compacts. Volume diffusion depends on three main paths. The first vacancy path is from the neck surface through the particle interior and onto the particle surface. This path results in mass transport to the neck surface since mass flows in the opposite direction of vacancies. There is no densification as the mass originates at the particle surface and is deposited onto the neck. German refers to this path as “volume diffusion adhesion” to avoid confusion with volume diffusion paths associated with densification [19].

The second volume diffusion path is the flow of vacancies from the neck surface to the interparticle grain boundary. This path results in densification as the vacancy is removed from the grain boundary where it is replaced by an atom. The result is the shifting of the center of mass of the particle towards the grain boundary and a decrease in the distance in the centers of mass of the neighboring particles. A decrease in the distance between centers of mass is one method of describing densification.

The third and final volume diffusion path is the annihilation of vacancies by dislocation climb. The vacancies are eliminated by the removal of the vacancy from the interior of the particle to the surface. The vacancy removal increases the density of the particle in the process.

Temperature controls the volume diffusion rate by determining the concentrations of vacancies in the particles. The concentration of vacancies is also controlled by the curvature of the particles and can be estimated as:

$$C = C_0 \left[1 - \frac{\gamma\Omega}{kT} \left(\frac{1}{R_1} + \frac{1}{R_2} \right) \right] \tag{13}$$

where C_0 is the equilibrium vacancy concentration,

γ is the surface energy,

Ω is the atomic volume,

k is Boltzmann’s constant, and

T is the absolute temperature, and

R_1 and R_2 are the radii of the particles [19].

As the grain sizes get smaller, the vacancy concentration departs more from the equilibrium concentration. The vacancy concentration is higher than equilibrium for a concave surface and less than equilibrium for a convex surface. The flow of mass is from the particle (lower vacancy concentration) to the neck (higher vacancy concentration) because the particle surface is convex and the neck surface is concave.

Volume diffusion is generally not as common as surface diffusion and grain boundary diffusion. The activation energy of volume diffusion is higher than the activation energies of the other two. Even though it is not as common, volume diffusion can occur late in sintering with undesirable results. As sintering progresses, the pores become isolated and almost spherical in shape. If two pores of different sizes are near each other, a vacancy gradient exists and the vacancies move from the small pore to the large pore until the small pore is eliminated. This results in one large pore and effectively coarsens the pore. Preventing grain growth is also important for conserving the grain boundary density; grain boundaries serve as the primary vacancy annihilation sites [19]. Densification can be achieved by the movement of pores to grain boundary sites.

c. Grain Boundary Diffusion

Grain boundary diffusion has an activation energy intermediate between surface diffusion and volume diffusion. Grain boundary diffusion is active due to the interfaces formed between sintered particles. As the neck grows, the grain boundaries grow and grain boundary diffusion becomes important. It is a contributing factor to densification of metals because the grain boundaries act as sinks for vacancies and result in the mass transport into the particle. This is particularly important for nanocrystalline compacts because the number of grain boundaries is much greater than for larger particles.

Sintering is limited if the grain boundary activation energy is too high. As sintering progresses, new grain boundaries are formed and the dihedral angle is high. A large dihedral angle is conducive to continued sintering and growth of the neck. As the dihedral angle becomes lower and the generation of grain boundaries becomes unfavorable, sintering is inhibited and can be described by the equation showing the balance between the neck size, X, and grain size, G, where:

$$X = G\sin\frac{\phi}{2}. \tag{14}$$

Once the equilibrium dihedral angle has been reached, any further neck growth is due to grain growth [19].

Finally, it is important to consider the role of grain growth when sintering materials. Grain growth occurs in metals at about half of the melting temperature due to the increase in grain boundary mobility [23]. In order to reduce the energy of the system, grains tend to form triple junctions, where three grains meet. The angle associated with the triple junction is 120°. At this angle, grains contain six neighbors and the energy is minimized. If a grain has fewer than six boundaries, each boundary is concave inwards and the grain will shrink. Grains with more than six boundaries will grow until the number of boundaries is six. Therefore, at elevated temperatures, larger grains will grow at the expense of smaller grains until the energy of the system is minimized. Grain growth must be considered because the sintering temperature and grain growth temperature are similar.

III. METHODS

A. KINETIC MONTE CARLO SIMULATION

The kinetic Monte Carlo (KMC) method is useful in simulating the kinetic and thermodynamic behavior of particles at a larger scale. The method can simulate grain growth and sintering on any spatial and temporal scale [21]. Because the KMC method is stochastic in nature, it is able to model processes that also behave stochastically, such as atomic diffusion or chemical reactions [21]. Instead of looking at the atomic level, the KMC method is able to model particles on the order of tens to millions of atoms, and to accomplish it on larger time scales. This thesis focuses on the utilization of the KMC method on grain growth and sintering because of its ability to model systems over a range of size and time scales. Particularly, this thesis is concerned with the ability of SPPARKS to adequately simulate the physics of grain growth and sintering.

Before continuing with the discussion of the operation of SPPARKS, the basics of MC simulations will be discussed. A basic understanding of MC methods is required to understand how SPPARKS works and to understand why KMC is suitable for simulating grain growth and sintering of particles.

1. Monte Carlo Ising Model

The most basic MC model is the Ising Model or two-state Potts model. The Ising model consists of a system containing two states. For simplicity, the two states considered will be described as spins of values zero and one. The basic premise behind the Ising model is whether a spin of one value will “flip” to the other value or remain unchanged. The probability of whether a spin will flip is based on several factors. When looking at simulations of grain growth, the flipping of a spin indicates that one grain is growing while the adjacent grain is shrinking. Consider a sight that has a spin of zero. This site is surrounded by eight nearest neighbors with like and unlike spins. Figure 10 shows the configuration of the sights.

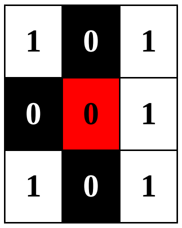


Figure 10. Ising Model Configuration With Eight Nearest Neighbors.

The site in the center of Figure 10 is a zero and it has three nearest neighbors with spins of zero and five neighbors with spin of one. There are two methods to determine the process of flipping the spin. The first method is Kawasaki dynamics where the site is randomly swapped with a nearest neighbor. If the site is swapped with a spin of zero, then nothing happens because the energy of the system does not change and the both sites remain at zero. If the site is randomly swapped with another site containing a one, then the configuration looks like Figure 11.

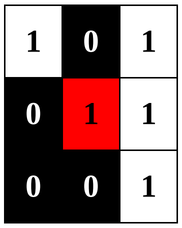


Figure 11. New Ising Model Configuration Using Kawasaki Dynamics.

The center site now has a value of one and the bottom left corner has a value of zero. The energy of the system (E) is given by the number of unlike neighbors for the center site and is given by:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{n}\left(1-\delta\left(q_i, q_j\right)\right) = \sum_{i=1}^{8}\text{unlike neighbors} \tag{15}$$

where N is the total number of sites,

n is the number of neighbors (8 for a two dimensional grid),

q_i is the state of the current site,

q_j is the state of the j-th neighbor site, and

δ is the Kronecker delta with $\delta\left(q_i = q_j\right) = 1$ and $\delta\left(q_i \neq q_j\right) = 0$ [24].

Equation 15 results in unlike neighbors contributing to the energy of the system, while like neighbors contribute no energy. The goal if KMC is to reduce the value of Equation 15. The change of energy in the system is calculated using the following formula [25]:

$$\Delta E = E_{final} - E_{initial} \leq 0 \, . \tag{16}$$

For this example, the energy of the system went from 5 to 4. Therefore, the energy of the system is reduced and the center site will flip to the new spin and remain a zero. The final configuration for the system will look like Figure 11. The Kawasaki model conserves the total number of each kind of spin (e.g., the number of spins with values of zero and one remain constant). The new configuration has produced a larger grain size of ones on the right side of the box. The number of connected sites with values of one went from three to four. Even though the grains appeared to grow, the Kawasaki dynamics is more a method for simulating ordering processes rather than simulating grain growth.

Glauber dynamics are better suited to simulate grain growth because the total number of sites with a particular spin is not conserved. This means that the overall system can increase in the number of zeros or ones instead of staying constant. Looking at the above example will show why this concept is important for grain growth. In Glauber dynamics, a site is chosen at random and then the spin of the site is chosen at random. If the site in Figure 11 is chosen to be a zero, then nothing happens because the site has not changed spins and the energy of the system is unchanged. If the site is given a spin of one, then the system becomes the configuration seen in Figure 12.

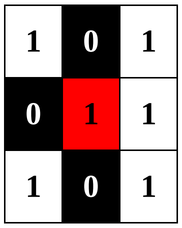


Figure 12. New Ising Configuration Using Glauber Dynamics.

The next step is to determine the change in energy. The original system had five unlike neighbors, while the new site has three unlike neighbors. Therefore, the energy of the system has gone down by two and the site remains a one. Therefore, the grain on the right of the box has become larger at the expense of the other grain. If the same site is chosen again and given a spin of zero, the energy of the system would go up and the site would remain a one.

The simulation of grain growth by the Ising model at finite temperatures requires the addition of a probability transition function. The probability transfer function accounts for the ability of a site to change spin values if the energy goes up. This ability to flip spins when the energy goes up is required to model the behavior of particles with an increase in temperature, thereby increasing the activation energy of the particle. The probability of a spin change is given by:

$$P(\Delta E) = \begin{matrix} 1 & if\, \Delta E \leq 0 \\ \exp\left(\frac{-\Delta E}{kT}\right) & if\, \Delta E > 0 \end{matrix} \tag{17}$$

where kT is the thermal energy of the simulation [25]. Therefore, if the thermal energy is greater than zero, there is a finite probability that a spin flip with a positive change in energy will be accepted. For example, if the change in energy is three and kT is set at 2, the probability of flipping spins is

$$P(\Delta E) = \exp\left(\frac{-\Delta E}{kT}\right) = \exp\left(\frac{-3}{2}\right) = 0.223 \tag{18}$$

This probability is compared to a random number generated between one and zero. If the probability of Equation 18 is greater than the random number, then the spin will flip. If the probability is less than the random number, the original spin will be retained.

There are several lattice types that can be used with the Ising model. The first distinction is whether the lattice sites are arranged in two dimensions or three dimensions. The above examples are lattice sites arranged in two dimensions with eight nearest neighbors. The number of neighbors can be varied from four to as many as is needed in two dimensions, but normally four or eight are used. If a two dimensional triangular lattice is chosen, then the number of nearest neighbors will be six. For three dimensions, the number of nearest neighbors can be six first nearest neighbors or twenty six first, second, and third nearest neighbors.

The appropriate choice of boundary conditions is another key aspect to KMC simulations. The boundary conditions are required for sites lying on the boundary of the simulation volume and to ensure continuity throughout the model. The sites on the edge of the simulated area will have fewer neighbors than the interior sites. One common boundary condition is the mirroring of the boundary. In this case, the boundary sites are mirrored and the site has the required number of sites. A popular choice of boundary conditions is the periodic boundary condition. In this case, the sites on the opposite boundary are wrapped to effectively create the sites on the outside of the boundary. This boundary condition is easy to implement. A variant of the periodic boundary condition is the skew-periodic boundary condition. It is used when simulating flat boundaries that have a non-perpendicular intersection angle with the boundary [25]. The edge of the simulation area is wrapped with an offset so the flat area lines up. This results in a continuous flat boundary.

The basic algorithms used to determine the outcome of the Ising Model are given in Figures 13 and 14. Figure 13 contains the algorithm for Kawasaki dynamics and Figure 14 contains the algorithm for Glauber dynamics. Both Kawasaki and Glauber dynamics algorithms allow for the array of sites to be saved as a snapshot. These snapshots can be viewed and analyzed during post simulation analysis. The choosing of sites and spins are random, and for large simulations run for long times, the number of random numbers generated can be large. If the random number generator has a low repeat signature or does not truly select the random numbers from a uniform distribution, then the system may generate patterns of behavior that are not based on the physics of the simulation [25].

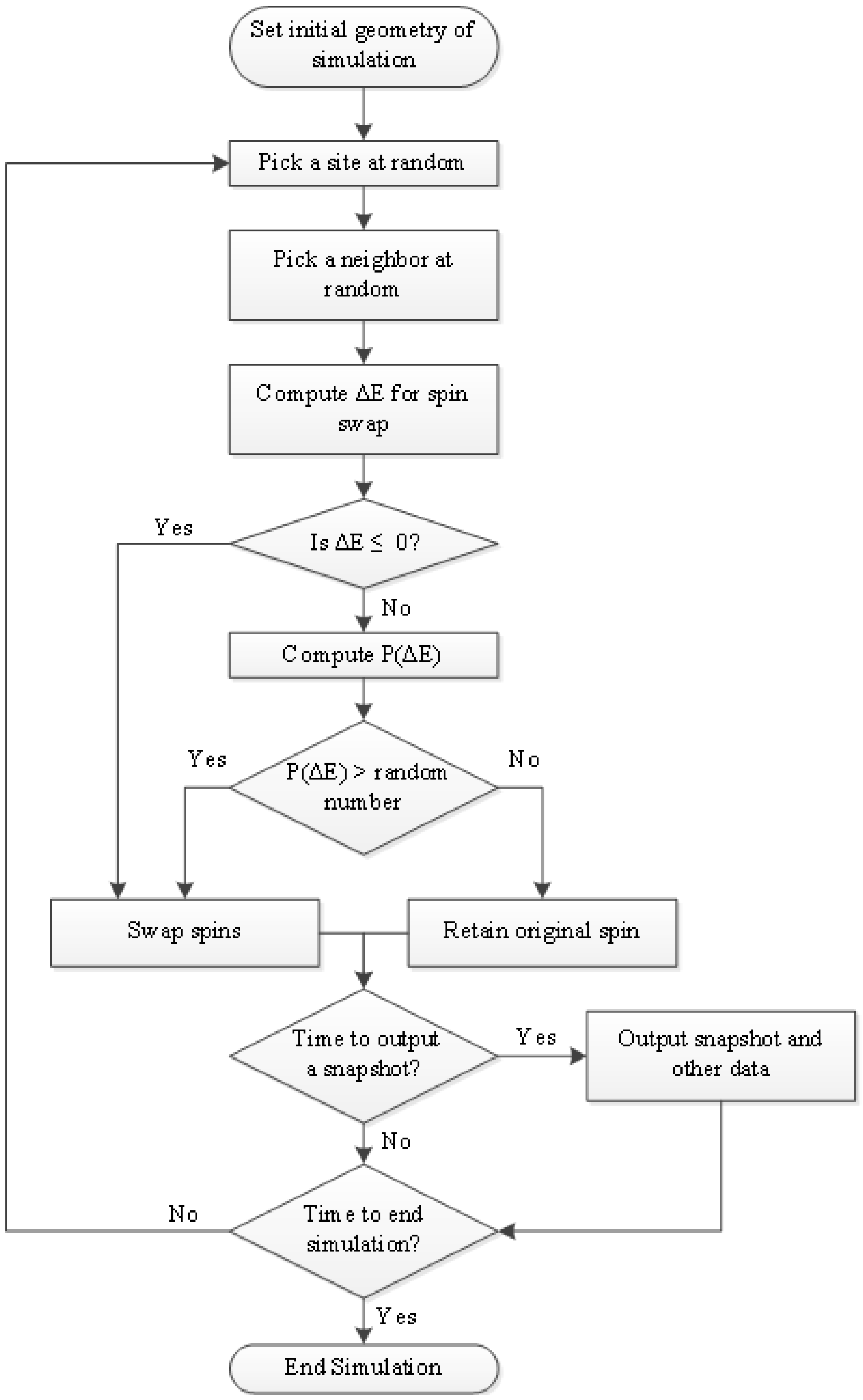


Figure 13. Basic Algorithm for Nonconserved Spin Ising Model Using Kawasaki Dynamics After [25].

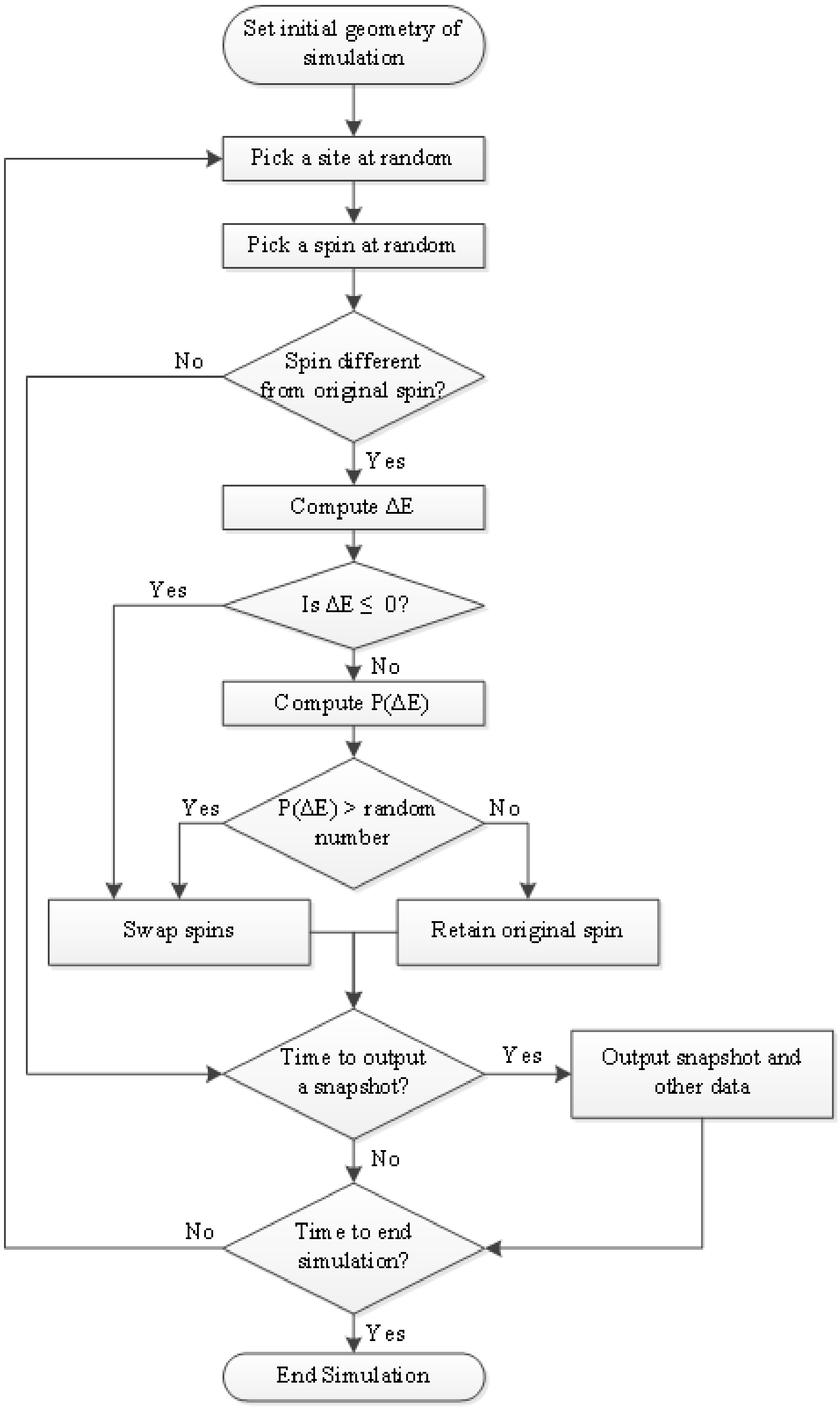


Figure 14. Basic Algorithm for Conserved Spin Ising Model Using Glauber Dynamics After [25].

2. Monte Carlo Q-State Potts Model

A more general case of the Ising model is the Q-state Potts model. The Potts model is almost identical to the Ising model with the exception that there are more than two spins. In fact, there can be Q states given for each site. Therefore, a site can have more values than zero and one, and each site can have a state value from zero to Q. This results in the model being called the Q-state Potts model. In the case of a polycrystalline material, the spins might be all of the possible grain orientations of individual crystallites, or grains, in the solid material. The Potts model can be used to simulate a much larger number of grains and is ideal for simulating grain growth and sintering. Each state can represent a different grain with a different orientation. The boundary between unlike spins is analogous to the grain boundary.

The following example shows how the Potts model can be used to simulate grain growth. Assume that there are ten states in the system ranging from one to ten. In order to simulate grain growth, Glauber dynamics will be used so that the spins are not conserved and the system allows grains to grow at the expense of neighbor grains shrinking. The first step is to randomly choose a site (Figure 15). The site in Figure 15 has one like neighbor and seven unlike neighbors, resulting in an energy of seven for the system.

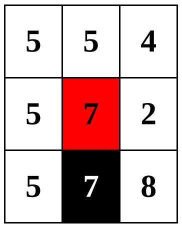


Figure 15. Potts Model Random Site.

Once the site is chosen, the spin of the site is randomly chosen from one to ten. In the first case, a value of three is assigned as seen in Figure 16. The site now has no like neighbors and eight unlike neighbors. The change in energy is given by the difference in the number of unlike neighbors in Figure 16 from the number of unlike neighbors in Figure 15; in this case +1 ($\Delta E = 8 - 7 = 1$).

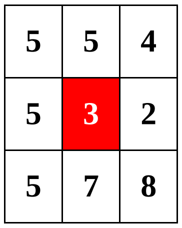


Figure 16. Potts Model New Site Value Using Glauber Dynamics.

Therefore, the energy of the system goes up and the site retains the old spin of seven. If a site value is chosen as two, four, seven, or eight, the change of energy in the system will be zero and the new spin will be accepted. For new spin values of one, three, six, nine, and ten, the energy of the system will go up and the old site is retained. If the kT value is set higher than zero, then there is a probability that one of these spin values could be maintained due to the thermal energy of the system. The final spin value that could be chosen is five as seen in Figure 17.

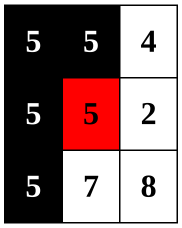


Figure 17. Potts Model New Site Value for Grain Growth.

For a value of five, the energy of the system goes from seven to four, a change of -3; and the spin value of five is retained. From Figure 18, it is evident that the grain with values of five has grown from 4 sites to 5 sites at the expense of the grain with value of seven. Of course, this example is extremely simple and in order to produce physical results, the number of sites and iterations needed must be must larger. The question is how large a simulation must be to get adequate results and are there algorithms that can run simulations large enough to produce realistic results.

The method for selecting sites and spins affects the kinetics of the simulation in an important way. The Potts model as implemented in SPPARKS normally uses Glauber dynamics and has three separate algorithms for choosing the new value of a sites spin. The first is the basic Potts model and a random site from 1 to Q is chosen. The next model, “potts/neigh,” chooses a spin randomly from the spins of neighbor sites and a null-bin, which extends the possible spins to all Q possible spins in the system. This algorithm weights the probability of picking a particular neighbor spin by the frequency of that spin in the current neighbor-set. If a site has eight neighbors with 4 different site values, then each neighbor spin will be chosen with a 1/8 probability and the range of spins from 1 to Q will be chosen with a 4/8 probability. The final model, “potts/neighonly,” discards the null-bin and will choose the four spin values with a ¼ probability [22]. The probability of swapping the selected spin based on its change in energy is the same for all three spin-selection models and follows the algorithm in Figure 14; however the kinetics of the simulation increases from spin-selection method one through three.

B. STOCHASTIC PARALLEL PARTICLE KINETIC SIMULATOR

This thesis utilizes the massively parallel program known as stochastic parallel particle kinetic simulator (SPPARKS). SPPARKS was developed at Sandia National Laboratories and is the first massively parallel KMC code [26]. As noted in the introduction, the ability to run extremely large simulations is key to simulating the evolution of nanostructures during grain growth and sintering. SPPARKS is an open source code that can run on a single processor or multiple processors. It is a parallel MC code for on-lattice and off-lattice models that includes algorithms for KMC, rejection kinetic Monte Carlo (rKMC), and Metropolis Monte Carlo. The code is written in C++ and is built by editing the makefile to produce and executable, such as spk_Hamming. Building the executable is determined by the platform running SPPARKS. This thesis focuses on the on-lattice, KMC code of SPPARKS. SPPARKS has many sub codes to facilitate a wide range of materials simulations. It contains algorithms for Ising and Potts modeling as well as others. This thesis will analyze the Sandia grain growth and sintering (SGGS) code developed for SPPARKS [24].

The SGGS code associated with SPPARKS contains algorithms for grain growth and sintering. The code can be run for grain growth, sintering, or both. Appendix B contains a sample input code for simulating the sintering of a cluster of particles. The input code sets the parameters SPPARKS uses to run the simulation. It contains a variety of inputs to the system and outputs for analysis of the model. The SPPARKS code annotates the time of the simulation in Monte Carlo steps (MCS). The MCS is a non-dimensional quantity and describes the step required to conduct a sweep of the simulation volume. Each model calculates the MCS and corresponding sweeps differently. For example, the SGGS conducts a sweep for every number of MCS based on the event ratios of grain growth, pore migration, and pore annihilation. The MCS is calculated as the total number of events divided by the number of grain growth events. In the case of the sample in Appendix B, each sweep takes a total of 3.5 MCS, where the total events is equal to seven and the grain growth events is equal to two.

The SGGS code for SPPARKS utilizes the Q-state Potts Model with Glauber dynamics. The Potts model is used because of the limitless number of spins. The spins can be different grain orientations of the same element or represent a variety of elements and phases. The designation of spin values is not important for determining the capabilities of the SPPARKS code. The Glauber dynamics are used because the spins must not be conserved in order to grow grains. The concept behind grain growth is larger grains grow as the smaller grains shrink. In order to adequately demonstrate grain growth, the spins must be able to flip without being conserved.

One important feature of the SGGS code and SPPARKS in general, is the use of temperatures. The temperature of the system is not an actual temperature, but rather an activation energy equal to kT (Equation 17). The activation energy is a used to generate a ratio between the energy of the system (number of unlike neighbors) and the activation energy to predict the probability of a spin swap. For example, a kT equal to one will result in an energy of the system as:

$$P = \exp\left(-\frac{\Delta E}{kT}\right) = \exp\left(-\Delta E\right) \tag{19}$$

where ΔE is the number of unlike neighbors.

The temperature command used in SPPARKS is analogous to the actual temperature of the system and can be determined by comparing the melting temperature to the temperature in SPPARKS where melting occurs. This comparison allows for a range of SPPARKS temperatures from room temperature to the melting point of a real system.

There are several key differences with the sintering code and the code for Potts grain growth modeling is the treatment of the boundaries. Potts modeling has several boundary conditions as described previously. These boundary conditions will not work for sintering because the continuous nature of the boundaries would preclude the densification of the particles and the movement of pores from the system. Therefore, the sintering code has solid boundaries (site value -1) and the interior of the model is a single simulation volume without a continuous boundary (Figure 18). This is one of the reasons for requiring large simulation volumes in order to have an adequate number of grains to simulate sintering.

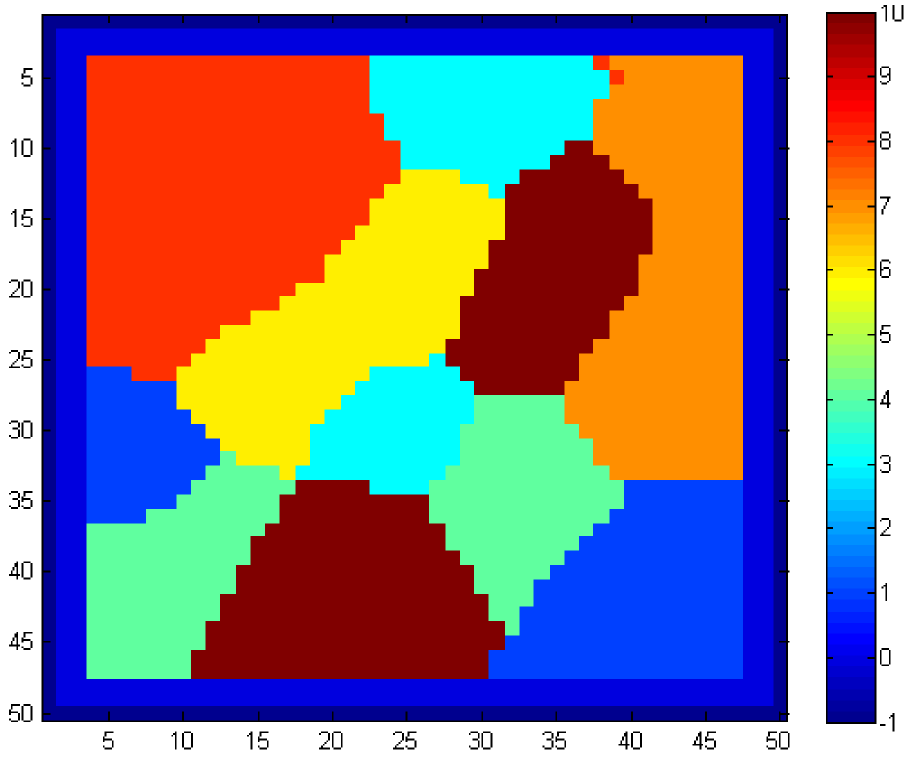


Figure 18. SGGS Simulation Box Using MATLAB.

Note: The dark blue pixels on the edge denote the boundary of the simulation volume (spin=-1). The lighter blue (spin=0) pixels denote open space.

The SGGS code contains other important features unique to sintering. As mentioned above, the goal of sintering is to densify the material from an initial green density. The SGGS code accomplishes densification by pore migration and pore annihilation. The migration of pores allows the pores to move from the interior of the simulation to the edges where the pore becomes part of the open space surrounding the simulation. The pore is replaced by a grain site and the density goes up. Density is calculated by a smaller cube inside the simulation volume. The cube in the SGGS code has sides with 1/3 the length of the sides of the simulation volume. This methods results in a cube with 33 sites per side for a 100^3 site simulation. Calculating density this way allows the pores to migrate out of the cube and result in density going up. Pore annihilation is performed by cancelling out two adjacent pores much like the annihilation of two dislocations interior to a grain. The SGGS code also has three distinct temperatures for calculating the probability of grain growth, pore migration, and pore annihilation. The grain growth temperature functions the same way as the Potts model. The pore migration and pore annihilation temperatures are very similar and determine the probability of the pore migration and pore annihilation events. Of course, there is only one physical temperature. It should be noted that the use of the word "temperature" in KMC simulations really refers to the thermal energy of the system, kT. Three separate temperatures really refer to three different activation energies for different physical processes.

C. NPS' HIGH PERFORMANCE COMPUTING RESOURCES: HAMMING

The simulations in this thesis made use of the Naval Postgraduate School's shared computing cluster, Hamming. As described earlier, these simulations are large, ranging from 10^6–10^8 lattice sites. Hamming is a Linux-based computing system with 1360 processors. Each processor has one gigabyte of RAM and the entire system has over 200 terabytes of storage. The Hamming computer system can be used to run parallel processes with OpenMPI. Useful information on Hamming can be found at http://hamming.uc.nps.edu/.

Hamming also has the ability to operate MATLAB, which was vital in producing microstructure figures from the output files. A MATLAB code, ‘SPPARKS_viewer,” was written to display the microstructure from the output file of site values. The MATLAB code inputs the site values and displays the microstructure in the x-y plane. The code has the ability to view the two-dimensional slices of the three-dimensional microstructure at any point in the z-axis. Figure 18 is a microstructure generated from the MATLAB code. The MATLAB code must be operated on Hamming because the dump files are too large to open on a personal or laptop computer. The raw file sizes range from 380 megabytes for a 250^3 simulation to 3.5 gigabytes for a 500^3 simulation. MATLAB can be run on multiple nodes and processors for files requiring more than one gigabyte of memory. In order to manage these large files, the dump files are written for a single time step as written in Appendix B under the “dump” command. Removing the “*” from the output dump file will write all time steps to a single file vice separate files. Having one file for smaller simulations may be useful, but for larger simulations (> 100^3), the file may be too large to easily find and view the data.

D. SIMULATION DEVELOPMENT

This thesis focused on a series of numerical experiments to determine the ability of SPPARKS to run large simulations. Many of the inputs were similar to the input file listed in Appendix B. The first step in running SPPARKS was downloading the software from the SPPARKS website (www.cs.sandia.gov/~sjplimp/spparks.html). Once downloaded, SPPARKS was compiled and a series of small simulations were run on a laptop (single four gigabyte processor) using the Ising, Potts, and SGGS models. After testing the operation of SPPARKS on a single processor, SPPARKS was downloaded onto Hamming and compiled with the procedures in Appendix A. The Potts and SGGS models were run on Hamming to verify the proper operation of the codes. The results generated on Hamming were identical to the results generated on the laptop computer.

Next, the ability to run large-scale simulations on Hamming using the SGGS model was conducted. Simulations ranging from 100^3 to 500^3 sites were run. These simulations were run to determine the simulation times, amount of RAM required, the minimum number of processors required to run a given simulation, and the size of the output “dump” files. After determining the efficiency and ability of Hamming to run the SGGS code, a simulation size of 250^3 sites was determined to meet the requirements of the thesis. In particular, this simulation size requires eight gigabytes of RAM and can be run on eight processors. The time to solve sintering simulations was on the order of several hours. These parameters allowed multiple simulations to be run at the same time on Hamming. MATLAB was used on a series of dump files to determine the size files that could be opened in order to view the simulation microstructures.

SPPARKS was also used to determine the operating characteristics of Hamming. The SGGS model was used to simulate a 250^3 problem on a number of processors ranging from six to 20 in order to determine the solve and communication times required. Hamming’s ability to run a series of simulations using the Potts model and SGGS code was examined. The Potts model was run with the “potts/neigh” and “potts/neigh” algorithms. The randomness of the SGGS model was determined by running identical 250^3 multiple time with the same “seed” value and five different “seed” values.

Once the ability to run simulations on SPPARKS using multiple processors was conducted, the ability of the Potts and SGGS models to accurately model grain growth was determined. The effect of temperature on grain growth was determined by varying the temperature from 1.0 to 6.0 using the Potts and SGGS models. Next, the simulation size was varied from 100^3 to 400^3 to assess the grain growth as a function of the number of sites using the Potts and SGGS models.

Several parameters were varied to determine their effects on the SGGS code. The porosity was varied from 5–40% to determine the effects of porosity on grain growth. Next, a 250^3 grain growth and sintering simulation was run to determine the accuracy of cluster size, cluster radius, and density. The grain size was also calculated to determine the rate of grain growth during both the grain growth stage and sintering stage.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND DISCUSSION

A. OBJECTIVE 1

Learn and set-up the SPPARKS code at NPS.

SPPARKS was successfully run on NPS' high performance computing system, Hamming. SPPARKS simulations were run to simulate grain growth using the Potts model and the SGGS code. Sintering simulations were successfully run via the sintering code. Using the procedures in the Appendices, SPPARKS can be loaded onto the NPS' high performance computer, Hamming. Once the SPPARKS code is compiled and the make file (binary) is produced, running simulations via SPPARKS is possible.

Appendices B through G contain the codes required to run simulations via SPPARKS and to display the results. Appendix B contains a detailed description of the executable code for SPPARKS. The example is for a sintering simulation containing 15,625,000 (250^3) sites with initially 30% porosity. Appendix C is an output of the sintering simulation in Appendix B. Appendix D is the procedure for generating an input microstructure to be read in by SPPARKS. For simulations larger than 200^3 (8,000,000) sites, MATLAB on Hamming should be used due to memory limitations on standard desktop computers. The procedure for loading and launching MATLAB on Hamming is contained in Appendix E. Appendix E contains the code for requesting the resources to run the simulation. The commands for executing this code are also located in Appendix F. Appendix G contains some useful commands for getting the status of jobs, deleting jobs, and editing large files that cannot be opened by other programs such as WordPad or text edit.

B. OBJECTIVE 2

Assess the performance of the SPPARKS code at NPS for large-scale simulation of grain growth and sintering of nano-scale materials.

1. Computational Performance

Quantitatively assessing the computational performance of the SPPARKS code is important to generating meaningful results in realistic time frames. Several parameters were evaluated to determine the performance of running SPPARKS on Hamming. Using SPPARKS ability to conduct simulations on parallel processors allows for much larger size problems than can run on a single processor or even a workstation with 4–8 processors. Running SPPARKS on a laptop computer with 4 gigabytes of RAM, allows for problems up to about 8,000,000 (200^3) sites. These size problems are too small to adequately simulate the physics of growing grains from the nano-scale to the micro-scale. Running on multiple processors should allow, in theory, for almost unlimited size problems. The limits are the required memory, which is finite, the speed of each processor, the communicate schemes between processors, and the methods for input-output (IO). Hamming has 1360 processors and can handle simulations over 125,000,000 (500^3) sites. In fact, setting up a one billion ($1{,}000^3$) site simulation required one terabyte of memory. Hamming was able to set up the problem using 512 processors (38% of Hamming's processors), but because the simulations required one terabyte of memory, 488 gigabytes of information was written to disk, which drastically reduced the efficiency.

In addition to varying the number of processors to run a simulation, the time to run a simulation is important. SPPARKS is able to conduct grain growth and sintering quickly, but as the simulations get larger, the time to complete the run necessarily increases. Using more processors can speed up the simulation, but there are a number of factors to consider when running increasingly larger simulations, such as outputting data and visualizing the microstructures. One factor to consider is the time to write the output files. The output files take approximately eight times longer to write for a 500^3 problem as opposed to a 250^3 problem because the size of the problem is eight times larger (125,000,000 vice 15,625,000). The differences in running different SPPARKS codes and the use of MATLAB were also examined. For example, the time to upload a microstructure for a 250^3 simulation was approximately 50 seconds compared to seven minutes for a 500^3 simulation, which results in a linear increase in processing time.

SPPARKS was able to run on Hamming with satisfactory results. The code could be further optimized for efficiency, but SPPARKS as written is able to run large simulations (e.g., 500^3 sites) on Hamming. Of course, obtaining resources on Hamming may be difficult if the demand is high and smaller problems may need to be run. Due to the size of the simulations, eight 250 cubed problems can be run for the same amount of memory and processors as one 500^3 problem. It is important to prioritize simulations to be able to generate the optimal results of grain growth and sintering using SPPARKS.

2. Number of Processors

Sintering simulations were run with a size of 250^3 sites. The number of processors for the simulation was varied in steps of two from 6 to 20. The simulation was run twice for each number of processors to determine the variation in run times. Appendix H contains the data and times obtained from the simulations. The total solve times varied from about 8,400 seconds (2.33 hours) for six processors to about 3,100 seconds (0.86 hours) for 20 processors. Figure 19 shows the total simulation time versus the number of processors used. The total times were consistent for the two runs and plateaued at about 14 processors before decreasing again as more processors were used. The plateau at 14 processors could be due to the number of nodes and processors used on Hamming and the arrangement of the processors on the nodes. Hamming assigns processors based on a queuing program and does not guarantee that all the processors will run on one node unless specifically requested. It could also be a property of the SPPARKS code in running the simulation as more than 16 processors are used. Most likely the stabilization of the solve time is a combination of the way Hamming and SPPARKS work. Variations in total run time could be caused by the configuration of processors used. For instance, using eight processors on one node is faster than running eight processors on eight separate nodes because the communication time increases. Other factors such as other processes running could influence the time required to run a simulation. In fact, the times required to actually run the solver were consistent and decreased with the number of processors used.

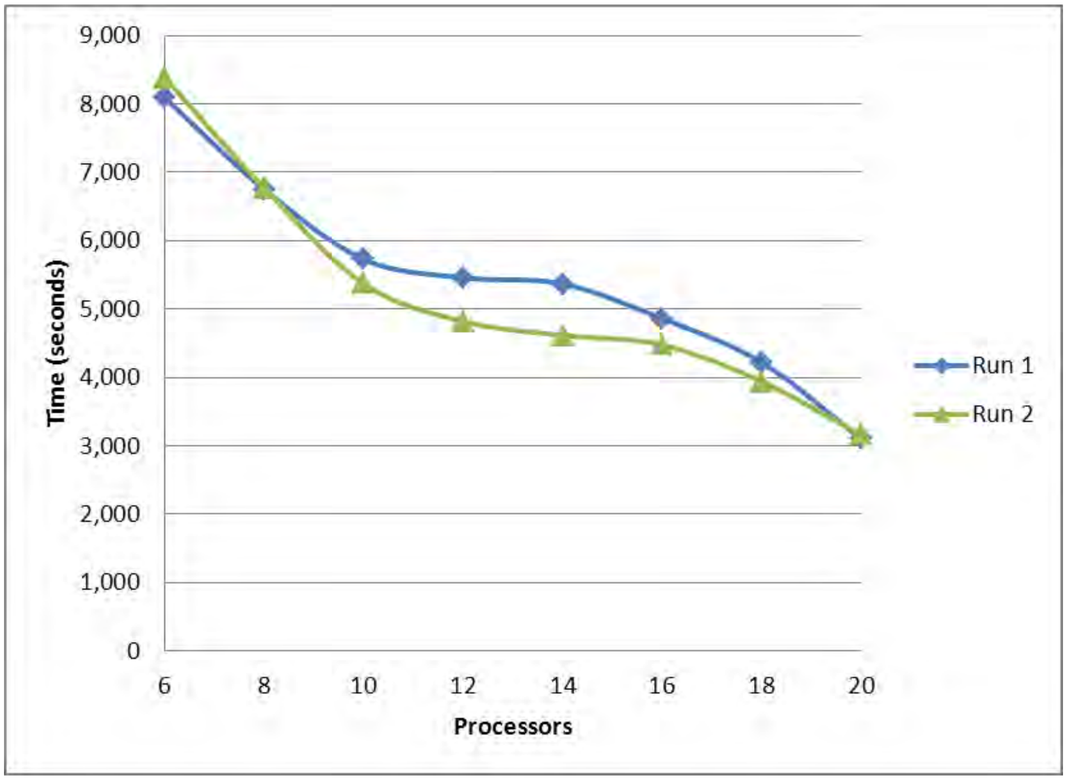


Figure 19. Total Simulation Time Versus Number of Processors.

The memory per processor is also an important statistic, as each processor on Hamming has one gigabyte of memory. Exceeding one gigabyte will cause other processors to pick up the excess and could cause degraded performance for the simulation or other jobs located on the same node. The total memory for the system generally went up for the number of processors due to the communication required across processors (Figure 20). The reduced memory required for 18 processors on run two may be a function of the Hamming system and location of the processors on the nodes. Other jobs running on Hamming could have an effect on the KMC simulations. As the number of processors goes up, the memory required to communicate site values from processor to processor goes up. Using processors on different nodes has little effect on the simulation time or memory. Requesting eight processors on one node vice the first eight nodes available may result in longer queue times if hamming is being heavily used.

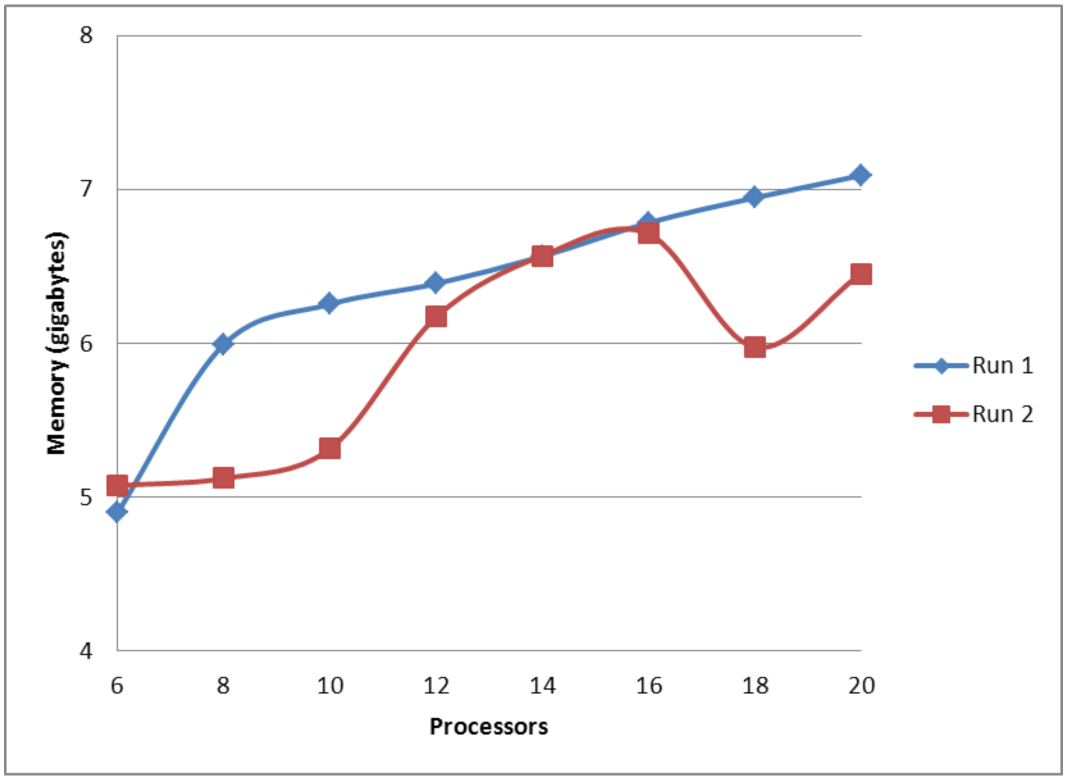


Figure 20. Total Memory Versus Number of Processors.

The final statistic important in determining the number of processors to use is the solve time. The solve time for the simulations went down by a factor of three (6,000 seconds to 2,000 seconds) from six to 20 processors (Figure 21). Even though the system was more efficient using a higher number of processors, using fewer processors may be advantageous. If hamming is being heavily used, requesting fewer processors would allow for more jobs to run at one time and may result in shorter wait times in the queue. Many of the simulations run with 250^3 sites for this thesis used eight processors with no noticeable degradation in performance. For KMC simulations, running simulations at about 50–75% of the processor memory is optimum for performance while minimizing the number of processors. The memory per processor is simply the total memory of the simulation divided by the number of processors. Exceeding one gigabyte per processor drastically reduces the efficiency and simulation time because the excess memory is written to disk vice the processor.

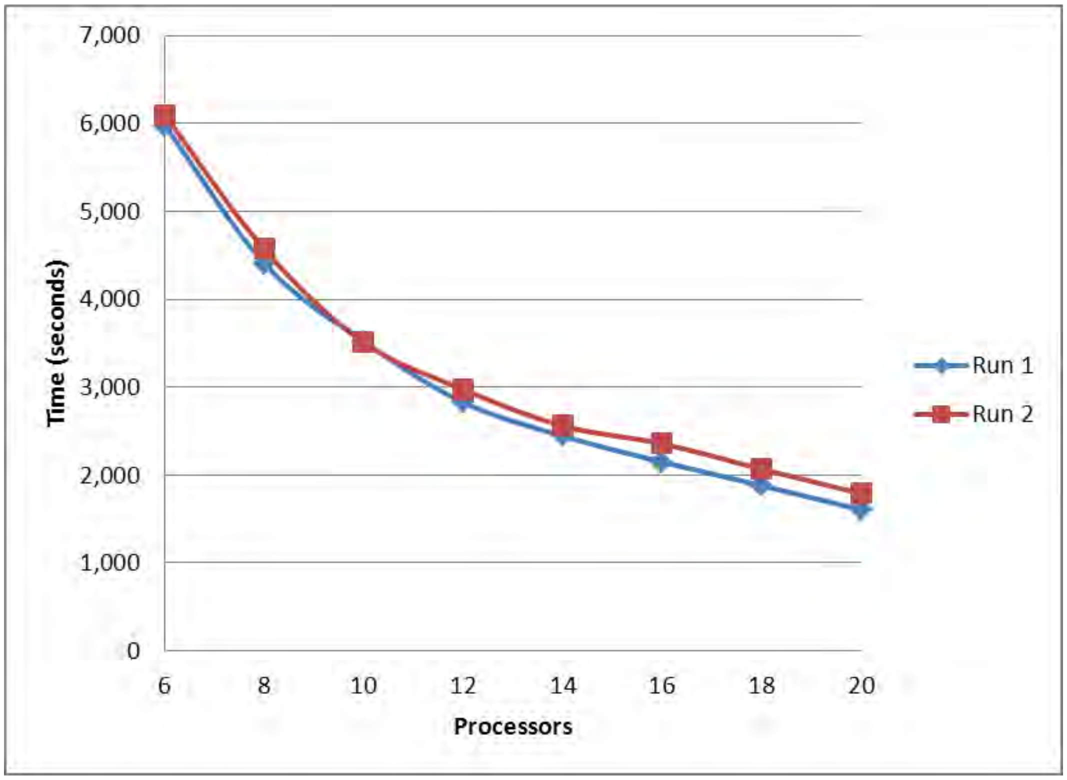


Figure 21. Simulation Solve Time Versus Number of Processors.

An interesting result from running the same simulation using a different number of processors is the final number of clusters and the values for N and R. Each simulation resulted in slightly different final values. The reason for the difference is most likely the 'seed' value which decides the randomness of the system. Each processor uses the 'seed' value to separately run the simulation. Therefore, running the simulation with 10 processors will result in different seed parameters than a simulation run with 14 processors because the simulation is divided differently. The results are identical for simulations run with the same number of processors and will be discussed in more detail in the next section.

3. Simulation Size Restrictions

The ability to run large-scale problems is key to modeling the physics of sintering at the nano-scale to the micro-scale. As mentioned in the introduction, a 500 sites per edge cube results in 125,000 10 nm grains. If the 10 nm particle are sintered to grain sizes of 100 nm, this will result in 125 grains in the simulation volume with 5 grains per edge. Reducing the sites per edge to 250 results in 15.625 grains/volume as seen in Equation 20:

$$\left(\frac{250\,sites/edge}{100\,nm/particle}\times\frac{1nm}{1site}\right)^3=\left(2.5\,particles/edge\right)^3=15.625\,particles/volume \quad (20)$$

It is apparent that 15.625 particles/volume may not be enough to adequately model the system and develop grain growth and sintering physics. To put this in perspective, the output file for the microstructure of 500^3 sites is on the order of 3–4 gigabytes vice only 350–500 megabytes for 250^3 sites. The microstructure file for a one billion site simulation is 29 gigabytes. In order to visualize the microstructures via MATLAB, the program must be able to open files of this size, which requires the use of Hamming. MATLAB on the current desktop computers in the MAE student computer laboratory are capable of opening a 200 megabyte file which is about 200^3 sites.

Based on the above numbers, the optimum size problem is roughly 500^3 sites. This number of sites allows for the evolution of 10 nm size particles to 100 nm size particles while still retaining the physics of the system. A 500^3 site problem requires about 80 gigabytes of memory, which translates to at least 80 processors. Running one simulation would require a little less than 6% of hamming's capacity (80/1360). In addition to the time required to run the simulation, the time required to output and visualize the microstructure is important. Writing the microstructure file takes on the order of 10–20 minutes. Creating a starting microstructure per the procedure in Appendix D takes several hours. The reason for the long time is a result of writing the input file in text format, which is the format required by SPPARKS. SPPARKS can output files in binary form, which will typically write out much faster, however, the binary format must be converted to text before reading into SPPARKS as an input file. In the future, SPPARKS should be amended to read binary input files which provide a substantial savings in system memory and time.

4. Performance Differences between Potts and SGGS Codes

One of the tests conducted to determine the computational performance of SPPARKS was to compare simulations between the Potts and SGGS codes. The Potts model uses both the “potts/neigh” and “potts/neighonly” algorithms for grain growth simulations. The algorithm choosing only spins of the neighbor sites is much faster than the neighbor algorithm. This is apparent because every spin chosen in the “potts/neighonly” algorithm has a higher probability of swapping spins and resulting in grain growth, whereas the “potts/neigh” algorithm has a lower probability of resulting in grain growth because random spins can be chosen that do not correspond with the neighboring spins. The grain growth for the sintering code is comparable to the nearest neighbor algorithm. The major difference is that the Potts model will ultimately result in one final grain if the simulation is run long enough. Depending on the initial Q value (number of spins in the system), the Potts model will take between 24 and 48 hours of simulation time to grow to a single grain for simulation sizes in excess of 250 sites/edge using the nearest neighbor algorithm. The simulation times and memory required are close for the Potts nearest neighbor model and the SGGS code sintering grain growth model.

The sintering portion of the sintering code does not run as fast as the grain growth portion. As an example, a 500^3 site problem conducted 400 grain growth sweeps in one hour 24 minutes. Once sintering began, the simulation slowed down considerably and the next 18 sweeps required six hours and 33 to complete. The longer time for sintering may be due to the requirements to track pore migration and pore annihilation, which requires more data to be stored and more communication between processors. The grain growth portion of the SGGS code The initial conditions and porosity are partially responsible for the delay in simulation time. Overall, the simulation of grain growth using the Potts model and SGGS is very similar.

C. OBJECTIVE 3

Identify and test key physical variables required to properly model grain growth and sintering using kinetic Monte Carlo codes such as SPPARKS.

1. Characterization of Stochastic Behavior in Coupled Grain and Sintering Simulations

The stochastic nature of KMC simulations SPPARKS is implemented in the "seed" command of the input file. The seed value defines the random number generator for the model and affects the starting microstructure and the progression of grain growth and sintering. Random numbers are used to determine whether the probability of swapping spin will occur or not. The random number is compared to the probability in Equation 17. If the probability is higher than the random number generated from zero to one, then the spin value is swapped. It is important to note that two simulations with identical initial conditions and <u>run with the same seed</u> will produce identical results to within numerical precision. If one wishes to sample different, stochastic pathways with the same set of initial conditions, then different seed numbers must be chosen. Appendix I contains the results of running two separate simulations with five different seeds. The simulations were run for 70,000 MCS at a temperature of 1.0. The difference between the sets of simulations was the porosity, which was set at 0% and 30%. Figure 22 illustrates the variation in R for the different simulations at 30% porosity starting at a radius of one voxel. The change seed value does introduce a small variation into the simulation results, but the overall trend remains the same. Due to the similarity among these results, running simulations with different seeds may not be worthwhile and comparing simulations with identical seeds is a way to remove the randomness from the system in order to isolate other parameters of interest such as temperature or density.

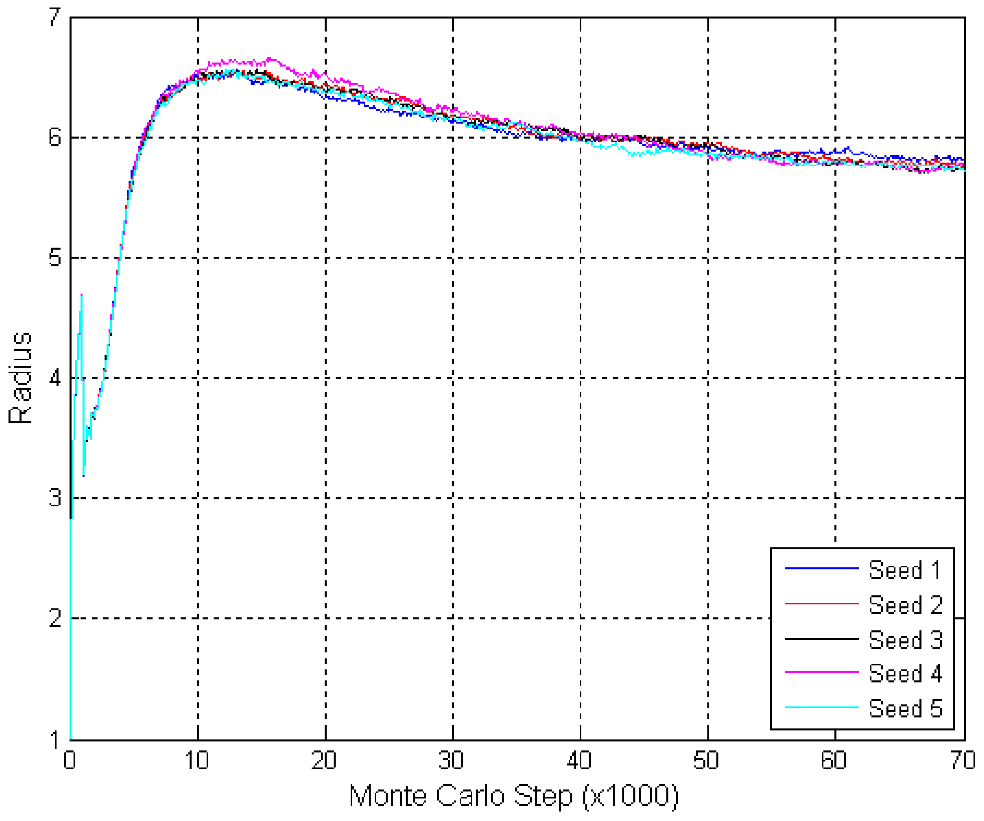


Figure 22. Grain Growth and Sintering as a Function of Seed Number.

2. Grain Growth as a Function of Temperature

Both the Potts model and SGGS code were run to determine the rate and characteristic of grain growth when varying the temperature. It should be noted again that the “temperature” in KMC is really a thermal energy that is compared with the activation barrier energy for a given process. The results are shown in Figures 23–25. The two Potts simulations were run with temperatures ranging from 1.0–6.0. All of the simulations were 250^3 and the Q-state was 1,000,000. Both the neighbor (potts/neigh) and neighbor only (potts/neighonly) were utilized to simulate grain growth. The simulations were run for 192 computational hours (24 hours x 8 processors). Figure 23 shows that temperatures 1.0 and 2.0 yield consistent grain growth to a radius of just under 25 voxels. The temperature of 2.0 has more fluctuations in radius (noise) due to the higher number of site swaps as the probability goes up, which results in more changes in radius. Temperature 3.0 has even more noise as the simulation progresses and peaks at a radius of 17 voxels. Proceeding to temperature 4.0, the radius rapidly increases to 6 voxels after about 1,500 sweeps and then stays constant. Temperatures 5.0 and 6.0 resulted in the same lack of grain growth. The fact that temperature 4.0 increases in radius and temperatures 5.0 and 6.0 do not are likely a computational artifact due to the high probability of energetically unfavorable transitions. Future work should be done to better tie the KMC temperature to the activation energy barriers for grain growth and sintering.

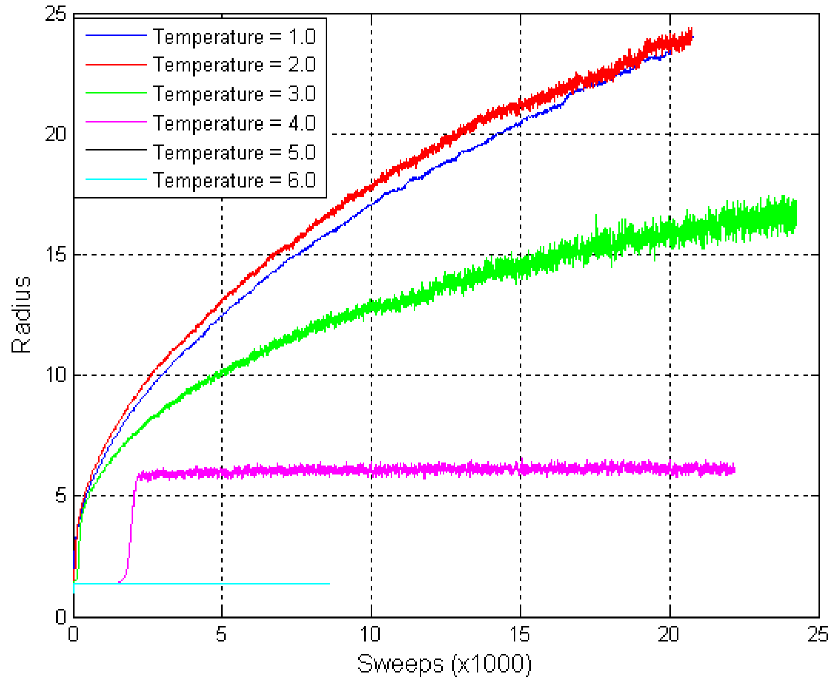


Figure 23. Potts Model Grain Growth Versus Temperature for Potts/Neigh.

The Potts model using the “potts/neighonly” also resulted in grain growth for temperatures 1.0–3.0 but with an increase in noise (Figure 24). The grain growth was more erratic and the fluxuations in grain size for these temperatures were even more sensitive to temperature. Of note, the fluxuations in the radii for temperatures 2.0 and 3.0 are much greater than for “potts/neigh.” The reason for the increase is the way the “potts/neighonly” chooses the site spin values as one of the neighbor site values.

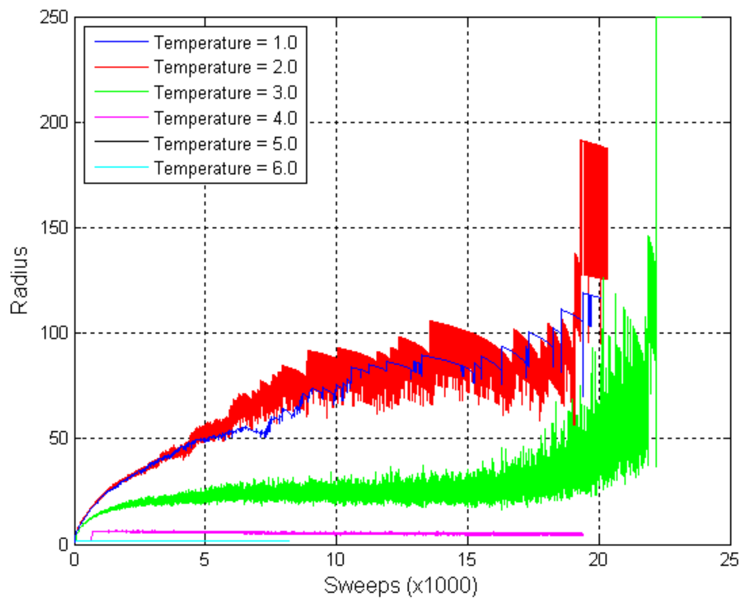


Figure 24. Potts Model Grain Growth Versus Temperature for Potts/Neighonly.

In addition, the "potts/neighonly" grains grow faster and the fluxuations are due to the rapid change in size of the large cluster surrounding two smaller clusters (Figure 25). The final result for temperature 3.0 is the formation of a single grain. Temperature 2.0 would most likely result in a single grain if the simulation was run longer. The resulting radii for temperatures 4.0–6.0 are identical to the "potts/neigh" radii and do not adequately model grain growth.

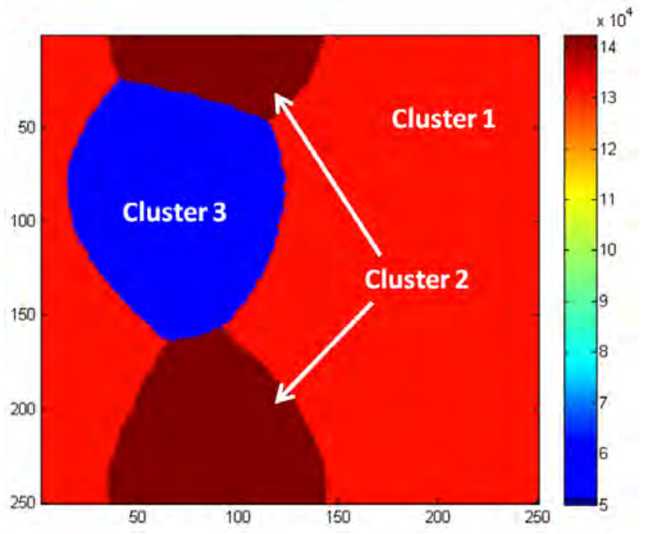


Figure 25. Potts Microstructure Showing Three Grains.

The final set of simulations performed were for grain growth using the SGGS code. The initial microstructure contained particles of radius one (single voxels), which is identical to the Potts model. The simulations were run for 70,000 MCS. The results of grain growth are shown in Figure 26. The grains grew initially in the temperature range 1.0–3.0. The grain growth for temperature 1.0 is fairly consistent, but for temperatures 2.0 and 3.0, the grain size appears to go down eventually. This is due to the nucleation of single voxel grains due to the increased temperature and higher probability that at spin can swap to a new value regardless of the energy change. As the Potts model revealed, the SGGS model does not adequately model grain growth for higher temperatures without compensation for the creation of single sites with new spins.

Overall, the ability to model grain growth via the Potts model and SGGS model does not reliably reflect the actual physics. In fact, as the temperature increases, the grains should grow more rapidly. Stated earlier, with 26 nearest neighbors, the melting temperature for both models should be at a temperature of 26. It seems that the models result in the physical onset of melting at much lower temperatures. The lack of apparent cluster growth is due to the formation of single voxel particles due to the increased likelihood of accepting a spin with 26 unlike neighbors from Equation 17. As the temperature goes up, the probability of forming a single voxel grain goes up until a temperature greater than 3.0 when the average cluster size does not go up.

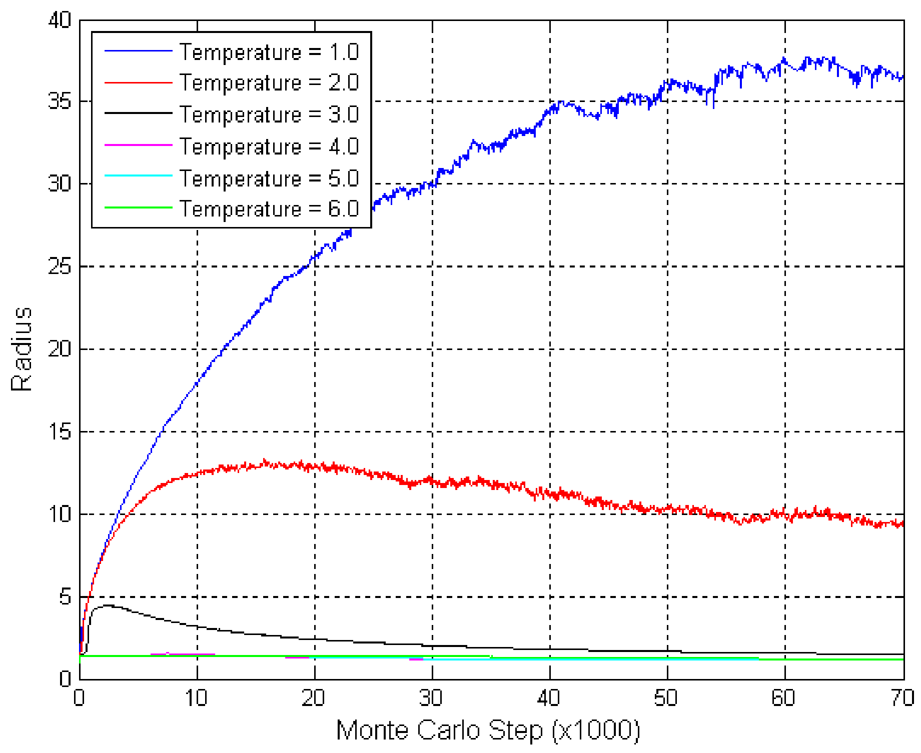


Figure 26. SGGS Code Grain Growth Versus Temperature.

3. Grain Growth as a Function of Simulation Size

The next set of simulations run were to examine the effect of simulation size (number of sites) on grain growth. The Potts model was used to grow the grains and are shown in Figures 27 and 28. Figure 27 shows the grain growth using the "potts/neigh" algorithm and Figure 28 shows the grain growth using the "potts/neighonly" algorithms. The temperature used in the simulations was 1.0 and all simulations were run for 24 hours of actual time with a computational time of 192 hours. The "potts/neigh" model results in consistent grain growth across simulation sites with higher noise at lower sizes. The grain growth was actually the same for all simulation sizes up to about 8,000 sweeps as seen in Figure 27. The "potts/neighonly" resulted in much faster grain growth as expected. The fluxuations in grain growth are due to the method used in the calculation of grain/cluster radius and the fact that as the simulation decreases to fewer grains and ultimately one large grain, the value for radius is less accurate. The jumps for 100 and 200 sites per edge are a result of simulation progressing to one grain. The 100^3 site simulation took 4,676 sweeps to reach one grain, and the 200^3 site simulations resulted in one grain after 11,918 sweeps. These results demonstrate the need to run large simulations capable of maintaining a minimum number of grains to adequately model the physics of grain growth. The simulations for 100^3 and 200^3 resulted in single grains fairly rapidly. Also, the fluctuations in grain growth decreased as the problem size was increased. In order to systematically estimate the minimum size, the simulation needs to be run at different sizes. The size above which the results are no longer changing appreciably with larger simulation sizes is the minimum size.

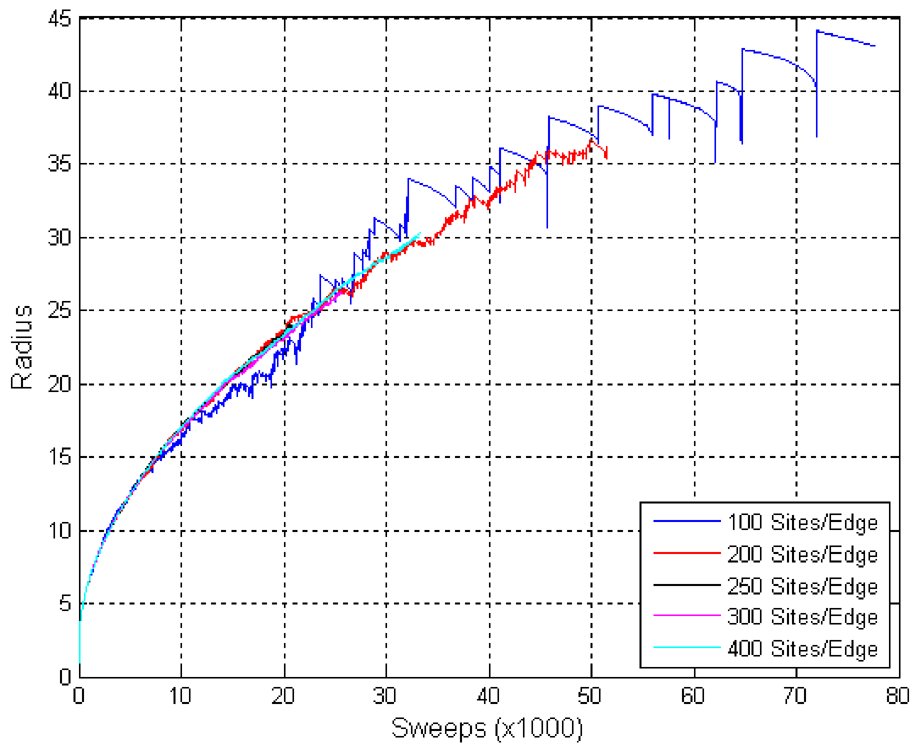


Figure 27. Potts Model Grain Growth Versus Size for Potts/Neigh.

Note: the curves in Figure 27 for the simulation sizes up to 8,000 sweeps are identical.

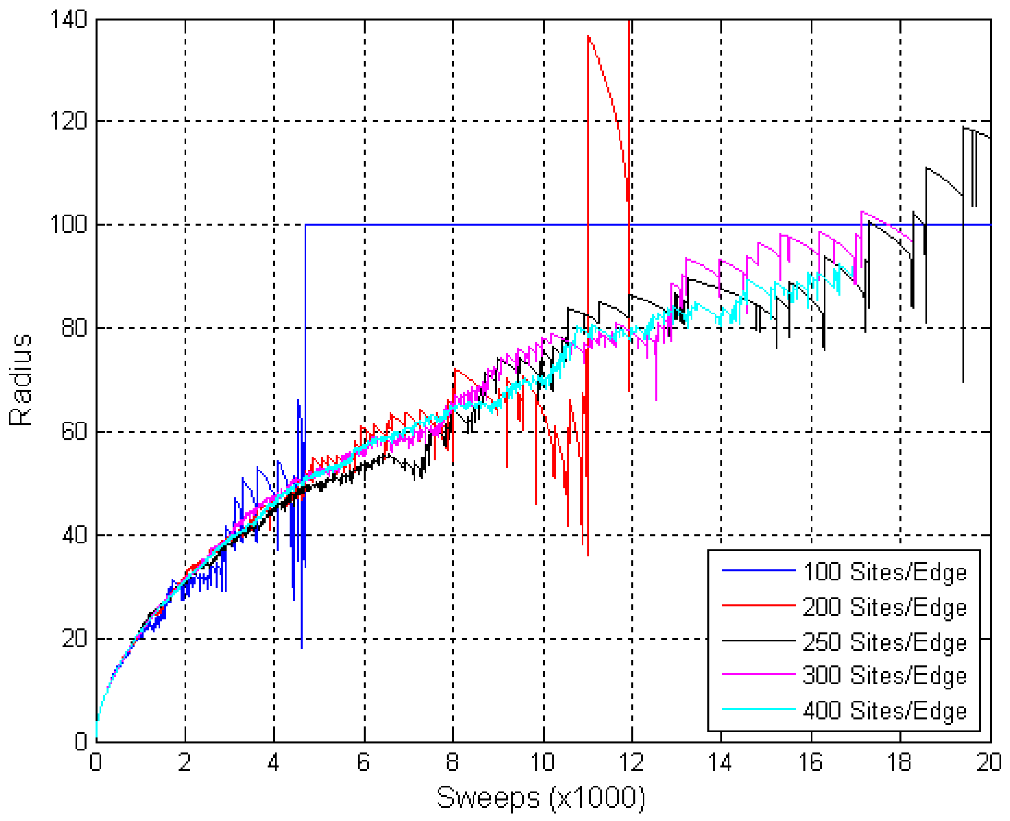


Figure 28. Potts Model Grain Growth Versus Size for Potts/Neighonly.

The SGGS model was also used to determine the effect of simulation size on grain growth and is shown in Figure 29. The temperature was set to 1.0, porosity was 0%, and sintering was turned off. The simulation was run for six simulation sizes and 210,000 MCS. The grain growth versus MCS is plotted in Figure 29 and the number of clusters versus MCS is plotted in Figure 30. The grain growth is greater for larger sized simulations. The interesting result is the reduced grain growth as compared to the Potts model. This is most likely due to the lack of periodic boundary conditions in the SGGS model induced by the boundary. The Potts model uses a continuous boundary to model the boundary conditions while the SGGS model has a wall at the boundaries. This may inhibit grain growth resulting in longer times to achieve a single grain, if possible. Figure 31 shows the continued grain growth for the larger simulations up to 350,000 MCS. Even though the grains continue to grow, they do not attain a single grain. In fact, the step increase in radius at 250,000 MCS is reminiscent of exaggerated or abnormal grain growth, and is most likely due to the conditions imposed by the boundaries on the simulation size and the dissolving of smaller grains as the larger grains grow. The SGGS model does an acceptable job in simulating grain growth and shows how large grains can be grown without porosity. In fact, all size simulations were similar in growing grains to a radius of 20 voxels.

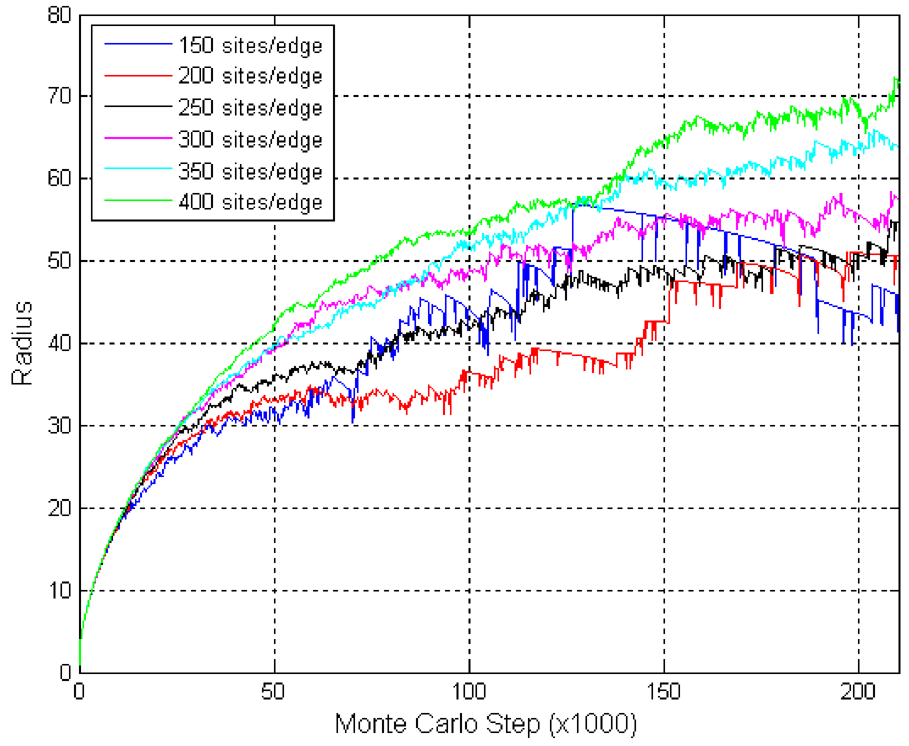


Figure 29. SGGS Code Grain Growth Versus Simulation Size.

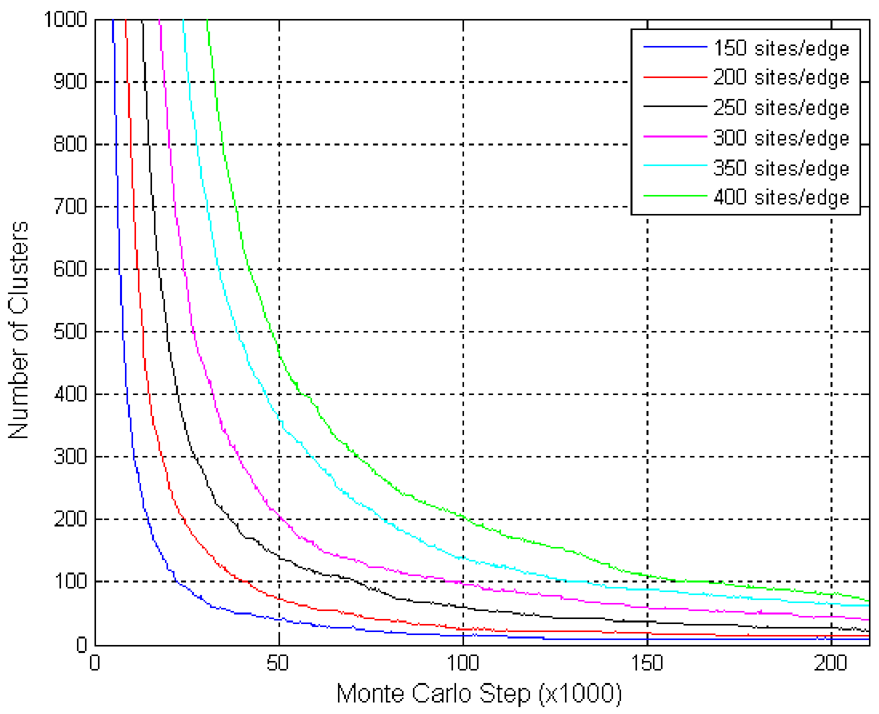


Figure 30. Number of Clusters Versus Simulation Size for SGGS Code.

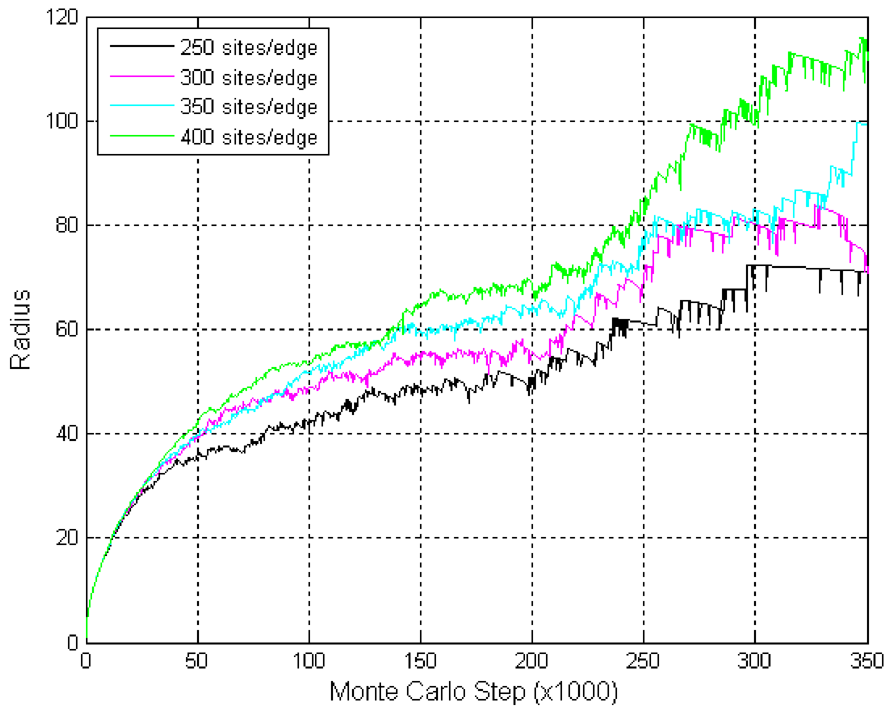


Figure 31. SGGS Model of Grain Growth Versus Simulation Size to 350,000 MCS.

4. Grain Growth as a Function of Porosity

The grain growth as a function of initial porosity is important for simulating the sintering of nanopowder compacts. These simulations also offer insight to how these phenomena are coupled in the SPPARKS-SGGS code. SPPARKS calculates N as the number of sites, or voxels, contained in each grain or cluster. The distribution of cluster sizes is given in Figure 32 for a 250^3 simulation with a temperature of 1.0 and porosity of 25%. The cluster sizes were taken at MCS 20,000 and appear to be represented by a log-normal distribution. Particle sizes are most naturally described by log-normal distributions as the minimum size must be zero. To get this result, it was necessary to remove the cluster sizes for the boundary (site values of -1) and the pores (site values of 0). SPPARKS does not remove these site values and uses them to calculate the value of N. For 25% porosity, the value of N calculated by SPPARKS is 1,114.7 while the value of N with the two site values removed is 778.9. Therefore, calculating the actual number of sites, or volume, of the grains is required to get an accurate representation of the microstructure.

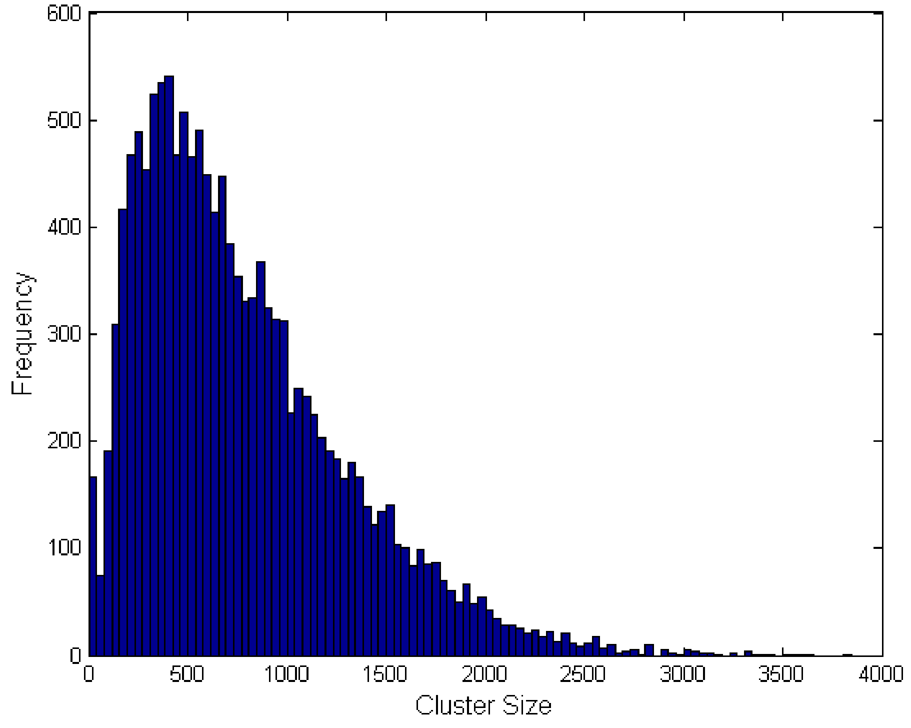


Figure 32. Histogram of Cluster Sizes for 25% Porosity at 20,000 MCS.

The distribution of pore volumes for a 250^3 grain growth is also log normal in nature. The sites per pore for a 5% initial porosity was calculated and the closed pore size was plotted at 20,000 MCS (Figure 33). The closed pores were plotted for 5% porosity because there were not enough closed pores to generate a log normal plot at 25% porosity (12,726 closed pores at 5% porosity vice 26).

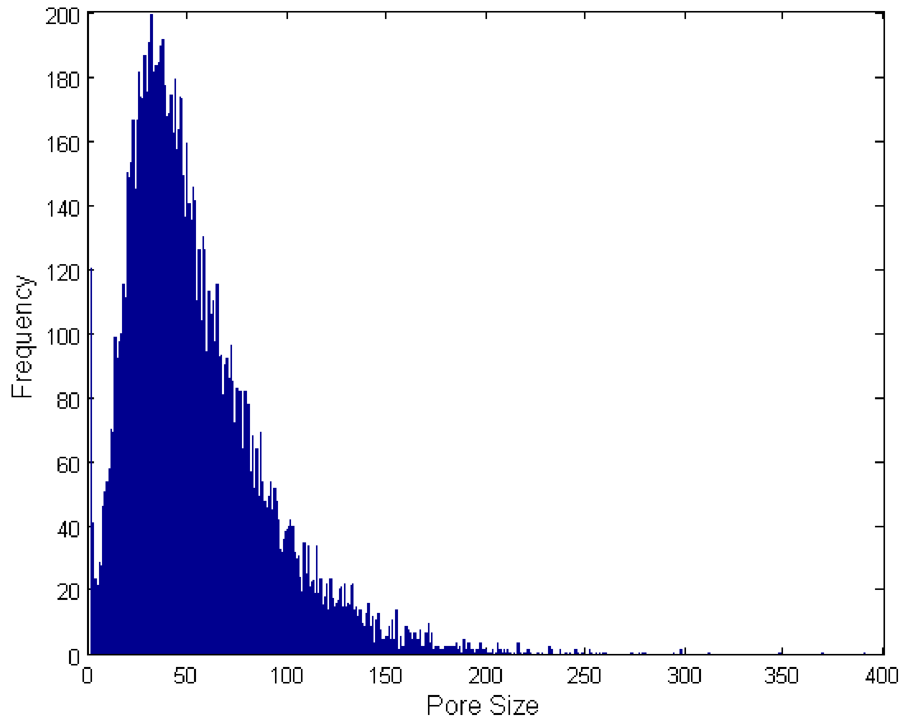


Figure 33. Histogram of Pore Sizes for 5% Porosity at 20,000 MCS.

The distribution of radii for the 25% porosity simulation was also calculated and plotted. In SPPARKS, the radius of each cluster is calculated by taking the cube root of the cluster size ($R = \sqrt[3]{N}$). The output radius is the average of the radii of the clusters. The distribution of the radius for a 250^3 simulation is plotted at 20,000 MCS in Figure 34. The distribution of radius does not follow a log normal curve but rather a Gaussian distribution. This change in distribution suggests that the current method for calculating R is not correct. R should also be log normal as is normally the case in grain size distributions. In the future, the SPPARKS-SGGS code should be amended to calculate R through a direct calculation from the Euclidean distance across the cluster. The peak at one in Figure 34 is due to the formation of the one voxel grains as described earlier.

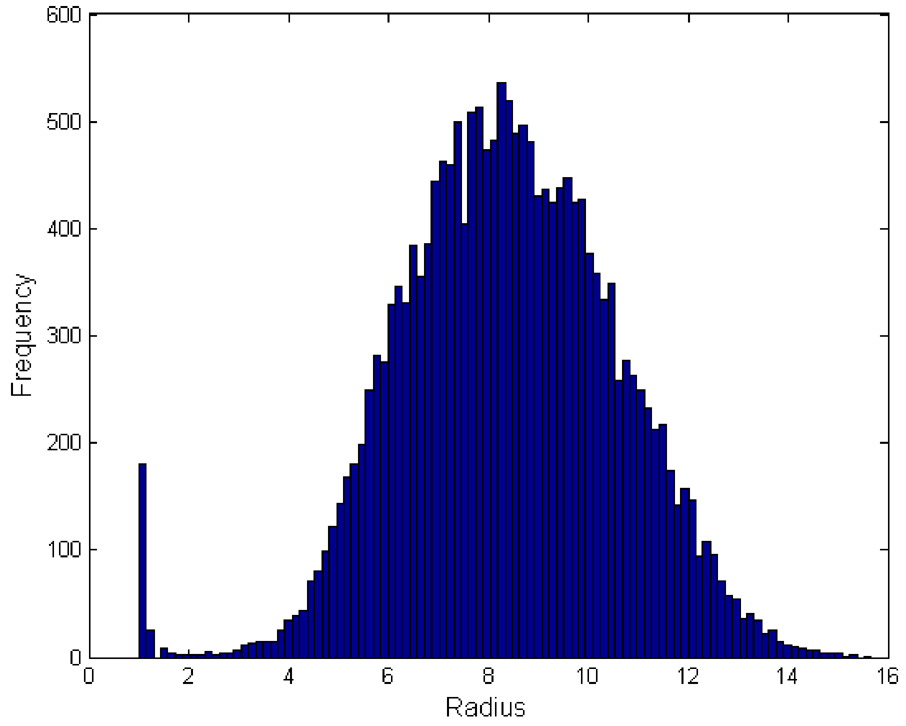


Figure 34. Histogram of Radius for 25% Porosity at 20,000 MCS.

The radius, R, also includes the clusters associated with the boundary and pores. In order to get the actual grain size, the clusters associated with site values of -1 and 0 must be removed. Figure 35 shows the grain growth as a function of porosity. At first glance, it appears that a larger grain size results at 15% initial porosity while the smallest grains are being grown at 5% initial porosity. This observations is counterintuitive because the lower the porosity should result in reduced pinning of grains and allow the grains to grow more rapidly. In order to accurately analyze the discrepancy, it is important to remember how SGGS calculates the cluster size and radius. The grain size (radius) includes all site values including -1 and 0. In order to get an accurate grain size, these site values must be removed. For a 250^3 simulation there are 372,008 sites in the boundary (spin = -1) and almost four million sites with value equal to 0. The results of these calculations are located in Appendix J.

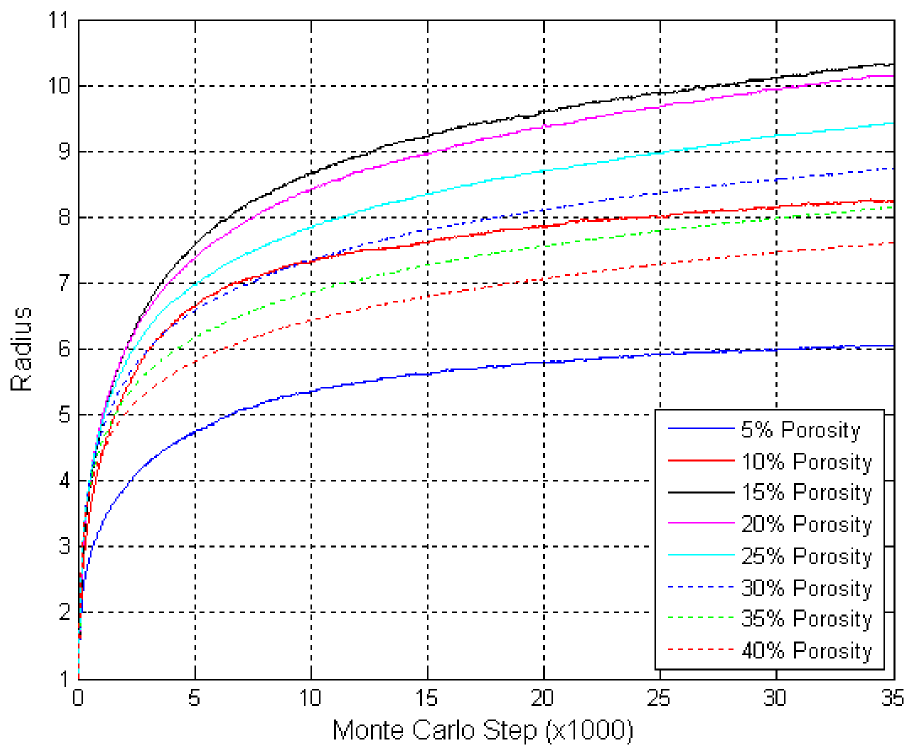


Figure 35. Radius Versus Time as a Function of Porosity.

The images from the simulated microstructures clearly show larger grains at lower porosity. Figure 36 depicts the microstructures for 5%, 15%, 25%, and 35% porosity. The dark blue areas are pores and are located at triple junctions and grain boundaries. As the porosity increases, the number of closed pores goes down and the pore structure is predominately open at higher porosities (Figure 36 C and D).

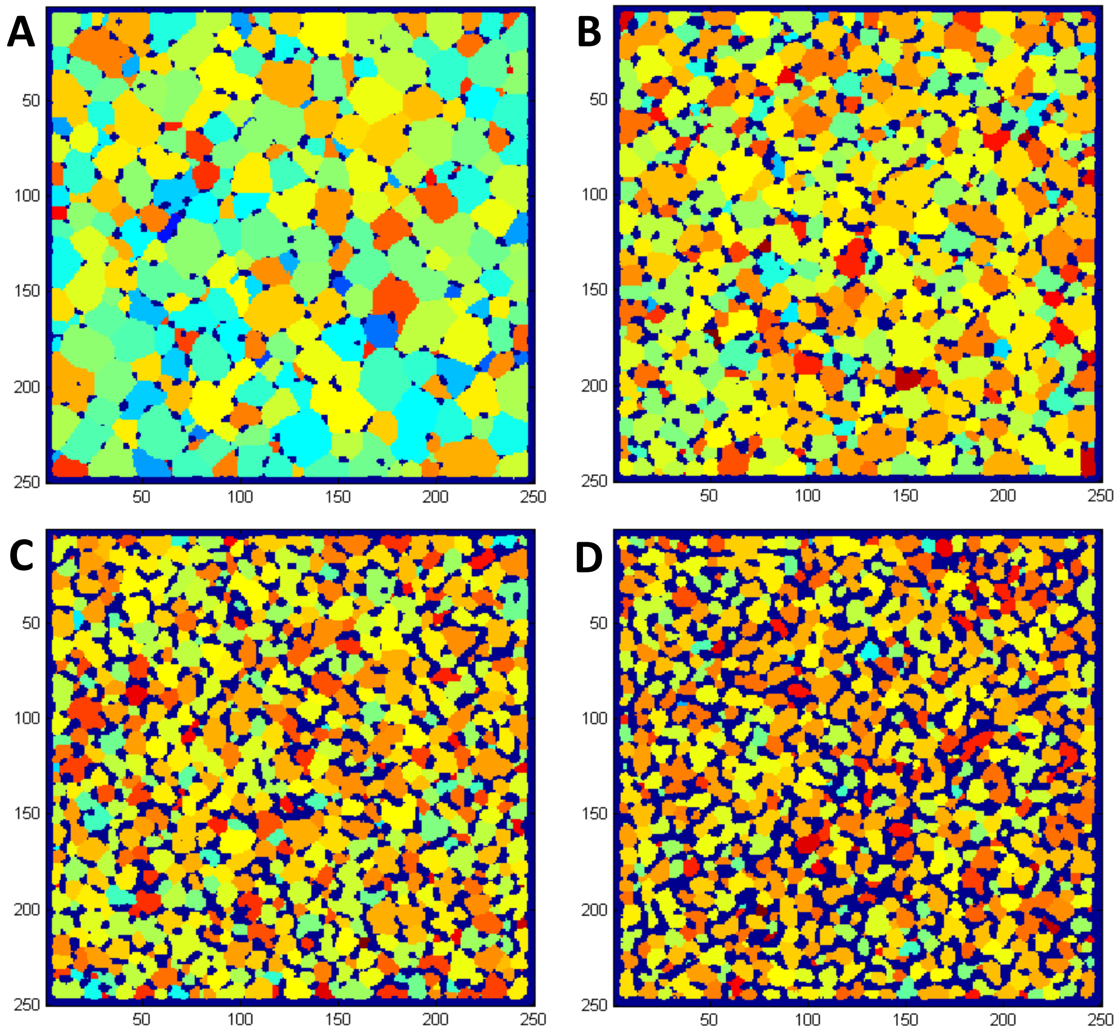


Figure 36. Microstructures of Grain Growth for Initial Porosities of A) 5%, B) 15%, C) 25%, and D) 35%.

Note: The different colors represent individual grains.

As stated earlier, the grain size is largest at lower porosity. Figure 37 shows the radius values calculated in SGGS and after removing the boundary and pores (both open and closed). The difference in radius is large for porosities less than 15%, but as the porosity is increased the SGGS and calculated radii of the grains are almost identical. For sintering, the radius is accurate at green densities of 60–70%, but becomes less accurate as the density approaches 90% as seen in Figure 37. From the porosity simulations, it is apparent that both the output data (density and radius) and the associated microstructures are important for understanding the physics.

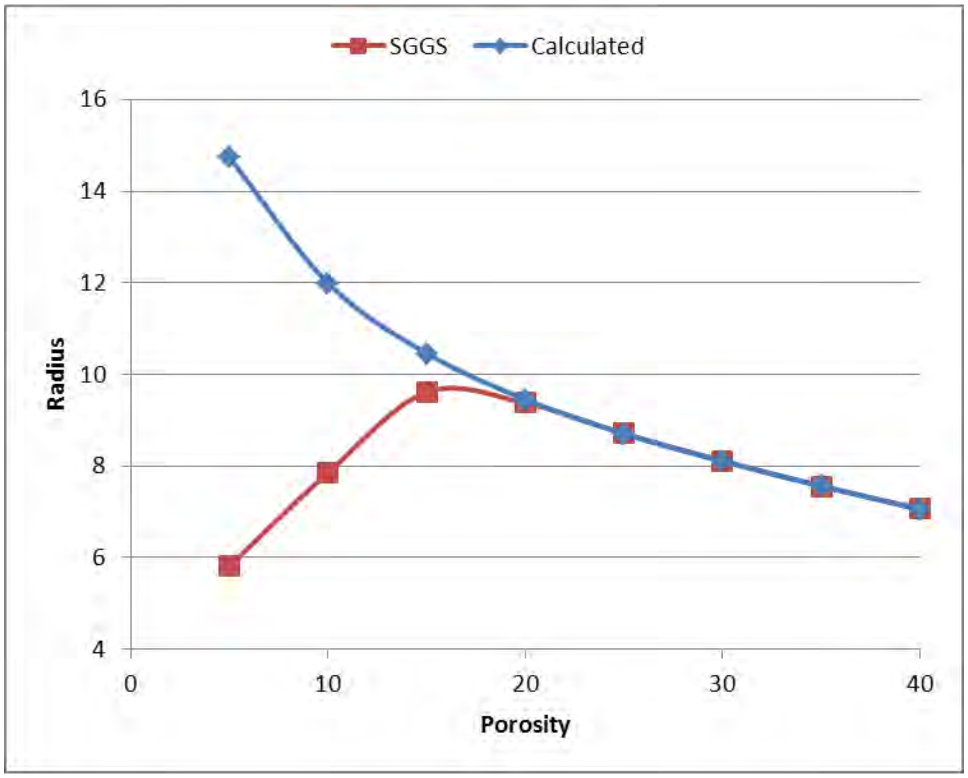


Figure 37. Radius at 20,000 MCS Versus Initial Porosity for Grain Growth.

5. Sintering Simulations

The SGGS code was used to simulate coupled grain growth and sintering. The simulation was run with grain growth to a radius of 4.68 voxels and a temperature of 1.0. The porosity was initially 30% and the simulations were run for 70,000 MCS. Sintering began at 980 MCS. Figure 38 shows the number of clusters and radius as a function of time (MCS). The number of clusters rapidly decreased until about 20,000 MCS when the decrease became linear. The radius initially goes from an initial grain radius of one to 4.68 and then drops immediately once sintering begins. This reduction in radius occurs due to the migration of pores and the initiation of small grains due to the temperature. SPPARKS allows grains to nucleate in the pores when the temperature is greater than zero. The radius then increases to about 6.5 before decreasing again. From the microstructures in Figure 39, it is clearly evident the grains actually grow throughout the simulation. The microstructure in Figure 39 (A) shows the microstructure at the beginning of sintering. It is not clear if the transition from A to B in Figure 39 is a real transition or a computational artifact.

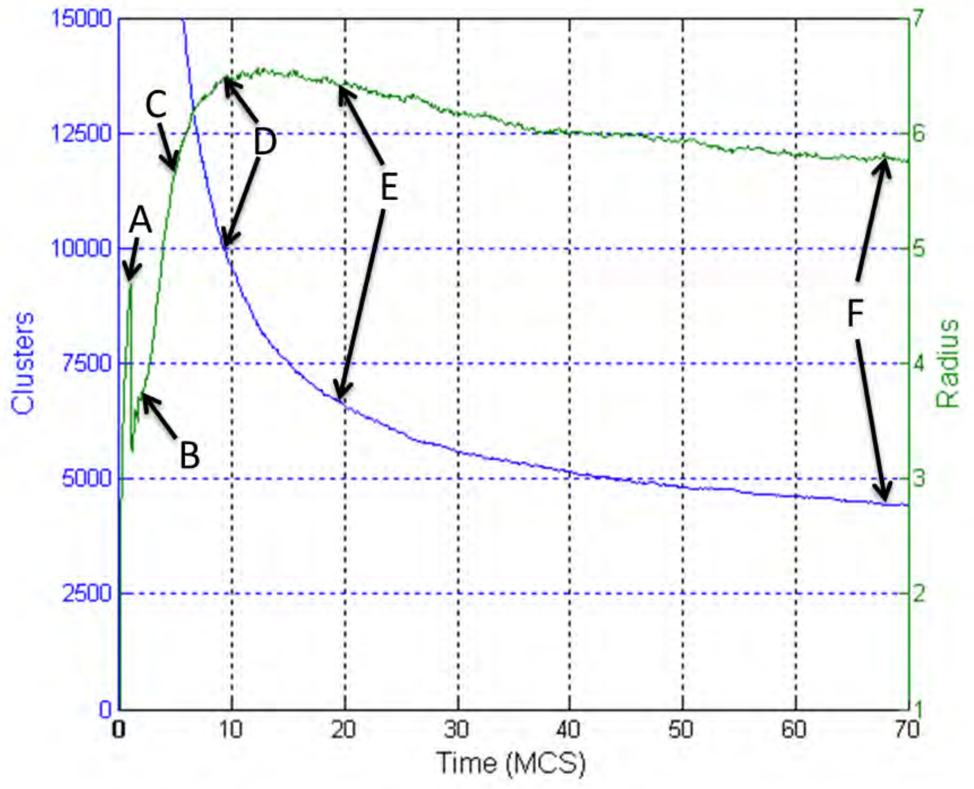


Figure 38. Clusters and Radius Versus Time for Sintering.

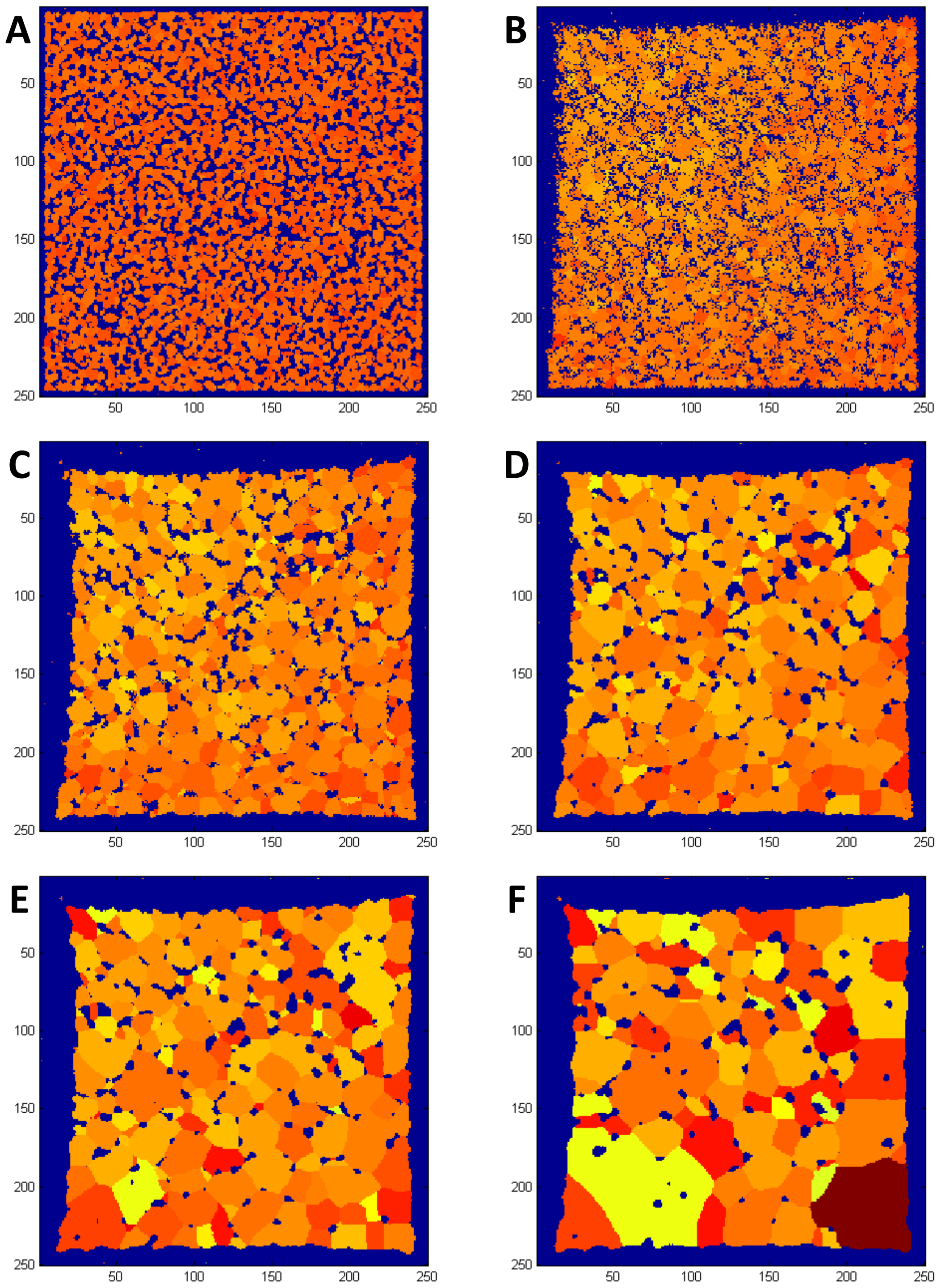


Figure 39. Microstuctures for Sintering at A) 980 MCS, B) 1,960 MCS, C) 4,900 MCS, D) 9,800 MCS, E) 19,600 MCS, and F) 68,600 MCS.

Note: All points in the figure refer to the alphabetic labels in Figure 39.

There are several important results from the microstructures in Figure 39. The first is that the pores roughen from A to B. This is analogous to pore migration and results from the pores moving from the center of the microstructure to the outside. In C, the pores are smoother and have begun to coarsen. In addition, the grains are growing slightly as sintering continues and the density increases. The pores have also migrated primarily to triple junctions and grain boundaries. As sintering continues further, both the grains and pores coarsen. Also, it is clear that the particle region is shrinking as the pores are accumulating at the top and bottom of the microstructures in Figure 39. From E to F, it is clear the grains have grown considerably while the density is fairly constant (Figure 40). In fact, there is a point in the simulation where the optimum sintering time is reached. At this point the density has begun to plateau and the grains continue to retain their nanoscale properties. After this point, the grains begin to grow and the material will lose its nanostructural properties. One reason for the lack of grain growth earlier in the simulation is the pinning effect of the pores. The pores prevent the grains from growing until the pores reach a minimum size and a larger inter-pore spacing. It is also important to note that the pores can become engulfed by the larger grains as sintering continues. Ideally, sintering would be halted and the temperature reduced once the desired density has been reached. In this case, the microstructure in Figure 39 D is optimum because the density is at 90% and the grains have not begun to grow appreciably and most likely have retained their nanoscale properties.

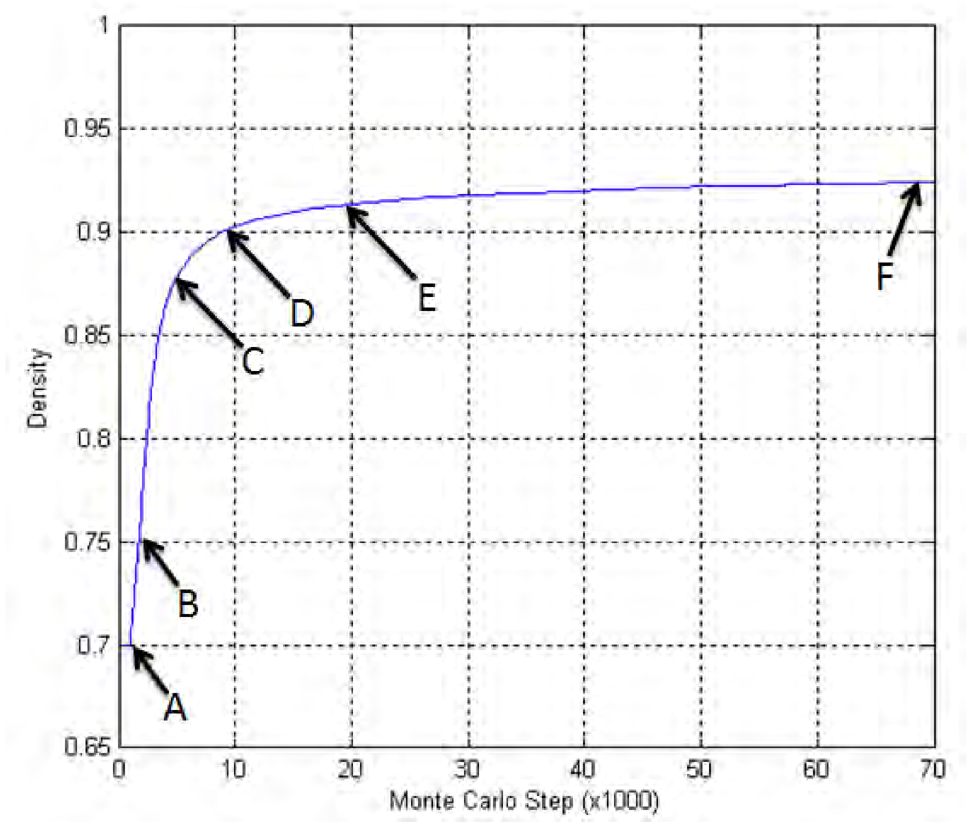


Figure 40. Density Versus Time for Sintering.

6. Sintering as a Function of Grain Size

Generating starting microstructures for sintering is important to systematically investigate the effect on different grain sizes on sintering. Unfortunately, SGGS does not have the ability to create microstructures of grains larger than 10 voxels when 30% porosity is required (Figure 41). As just discussed, the porosity effectively pins the grain boundaries, thus suppressing grain growth. The ability to generate a starting microstructure with larger grains is important to understanding sintering as a function of grain size and temperature.

Another strategy for growing grains larger with 30% porosity is to grow the grains with 0% porosity and then using this microstructure as an input to SPPARKS. Porosity can be assigned to this new, large-grained, input microstructure at a specified level, e.g., assigning 30% of the sites a spin of zero. Figure 41 shows the initial microstructure (left) and the microstructure after reaching a radius of 15. The porosity is randomly distributed throughout the microstructure and coarsens as the SGGS code is run. The grains themselves do not appear to grow, which results in an unrealistic starting microstructure on the right. In fact, what has happened in this simulations is that the pores have migrated without any grain growth in order to reduce the pore-grain interfacial area. The microstructure in Figure 41 B is reminiscent of pore migration and coalescence during high temperature irradiation of nuclear materials [27]. The ability to accurately produce an initial microstructure with 30% porosity is difficult and vital to simulating sintering using the SGGS code. Future work should include the creation of a code specifically for generating starting microstructures with specified grain size and porosity.

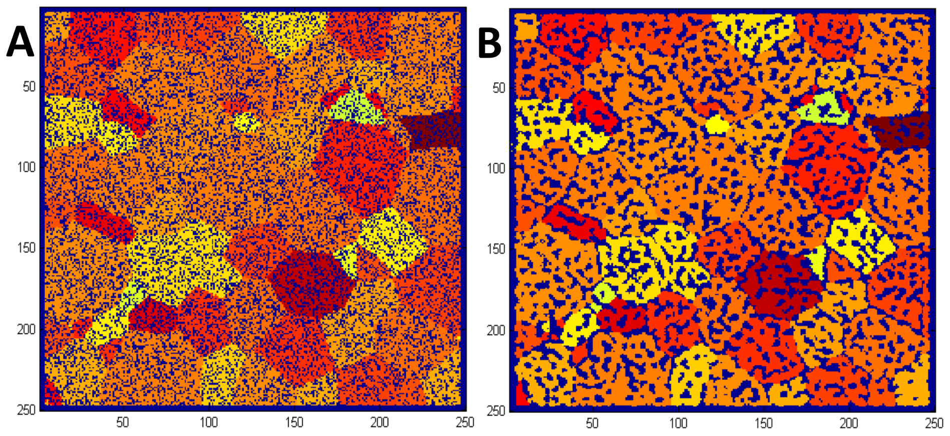


Figure 41. Microstructure with 30% Porosity Added After Grain Growth.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RECOMMENDATIONS FOR FUTURE WORK

Throughout the research and implementation of SPPARKS, specific aspects of the code that need to be addressed were identified. The most important area for improvement is the ability to create microstructures with both a specified grain size and porosity level. As of now, it is not possible to grow grains with 30% porosity greater than a radius of 9 using the SGGS code. Several approaches were used, but none produced a satisfactory structure. One recommendation would be to randomly input seed sites separated by the desired grain diameter apart and grow the grains from these sites. Once the grains come in contact, the grain growth would stop growing and the resulting microstructure would contain grains of the desired size with a green density between 65–75%.

Another area for future work is to have SGGS output the key parameters needed to describe grain growth and sintering. The first parameter is the value for grain size. SGGS calculates and outputs cluster radius, but it does not exclude the boundary sites and pores in this calculation. Other key parameters that would be useful in developing an accurate model for sintering are average neck size and average distance between centers of mass of the grains.

Finally, future work needs to address the effects of KMC temperature on grain growth and sintering. The correct scaling of the KMC temperature with the interfacial energy that drives the grain growth and sintering is not well understood and comparing the melting temperature of a simulation with 26 nearest neighbors does not correspond with a temperature of 26.0. In conjunction with a better scaling of the simulation temperature, the algorithm should be revised to exclude the nucleation of single pixel grains of new orientations as this phenomena is not a part of the physics of the grain growth-sintering problem.. A method of rejecting spin swaps resulting in single voxel grains should be implemented similar to the “potts/neighonly” algorithm.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

This master's thesis has investigated the ability to simulate coupled grain growth and sintering with SPPARKS and the Potts and SGGS codes. These codes were successfully installed and implemented on NPS' high performance computing resources. Compiling and running simulations on Hamming was performed and the ability to attain useful data and view resulting microstructures was achieved. Large input and output files (on the order of several gigabytes) were used by SPPARKS and viewed via MATLAB on Hamming using new MATLAB code written for microstructural data visualization.

The computational performance of the SPPARKS code on Hamming was assessed, and properties such as simulation time and memory usage were collected. Using multiple processors systematically decreases the simulation time as did running multiple processors on the same node. Ensuring the one gigabyte per processor is not exceeded results in faster simulation times because the system does not write to disk. Simulations sizes up to 500^3 can be successfully run on Hamming. Larger size problems take more time and the output files become impractically large. A 1000^3 simulation uses over one terabyte of information and takes up an unacceptably large portion of Hamming's resources.

The SGGS code can be used to accurately describe grain growth and sintering. The densification of the particles can be predicted and is a key parameter in assessing the sintering of grains at the nanoscale. The evolution of the microstructure is also accurately represented in the SGGS code and the grain size is consistent with the microstructure. The ability to determine the optimum sintering time and temperature can be determined using the SGGS code by analyzing the density, grain sizes, and microstructures. The SGGS code indicates that pores retard grain growth, as higher porosity results in smaller grains. In addition, simulating grain growth and sintering with the SGGS code illustrates the evolution of the pore microstructure and suggests that there is an optimum sintering time with respect to maximizing density while minimizing grain size.

The SGGS code does not have the ability to produce microstructures with grain radii greater than 10 voxels with 30% porosity. The SGGS code does not output average grain size, average neck size, or the average distance between grain center of masses. These parameters would be useful in accurately modeling and assessing the sintering of particles at the nanoscale.

LIST OF REFERENCES

[1] Z. Z. Fang and H. Wang, “Densification and grain growth during sintering of nanosized particles,” *International Materials Reviews,* vol. 53, pp. 326–352, Nov 2008.

[2] E. Ide, S. Angata, A. Hirose, and K. F. Kobayashi, “Metal-metal bonding process using Ag metallo-organic nanoparticles,” *Acta Materialia,* vol. 53, pp. 2385–2393, May 2005.

[3] M. Obrien, C. R. Rice, and D. L. Olson, “High-strength diffusion welding of silver coated base metals,” *Welding Journal,* vol. 55, pp. 25–27, 1976.

[4] E. Ide, A. Hirose, and K. F. Kobayashi, “Influence of bonding condition on bonding process using Ag metallo-organic nanoparticles for high temperature lead-free packaging,” *Materials Transactions,* vol. 47, pp. 211–217, Jan 2006.

[5] M. Maruyama, R. Matsubayashi, H. Iwakuro, S. Isoda, and T. Komatsu, “Silver nanosintering: A lead-free alternative to soldering,” *Applied Physics a-Materials Science & Processing,* vol. 93, pp. 467–470, Nov 2008.

[6] Y. Morisada, T. Nagaoka, M. Fukusumi, Y. Kashiwagi, M. Yamamoto, and M. Nakamoto, “A low-temperature bonding process using mixed Cu-Ag nanoparticles,” *Journal of Electronic Materials,* vol. 39, pp. 1283–1288, Aug 2010.

[7] M. A. Meyers and K. K. Chawla, *Mechanical Behavior of Materials,* 2nd ed. Cambridge; New York: Cambridge University Press, 2009.

[8] G. W. Nieman, J. R. Weertman, and R. W. Siegel, “Mechanical-Behavior of Nanocrystalline Cu and Pd,” *Journal of Materials Research,* vol. 6, pp. 1012–1027, May 1991.

[9] A. D. Albert, M. F. Becker, J. W. Keto, and D. Kovar, “Low temperature, pressure-assisted sintering of nanoparticulate silver films,” *Acta Materialia,* vol. 56, pp. 1820–1829, May 2008.

[10] P. Buffat and J. P. Borel, “Size effect on melting temperature of gold particles,” *Physical Review A,* vol. 13, pp. 2287–2298, 1976.

[11] M. Dippel, A. Maier, V. Gimple, H. Wider, W. E. Evenson, R. L. Rasera, and G. Schatz, “Size-dependent melting of self-assembled indium nanostructures,” *Physical Review Letters,* vol. 8709, pp. art. no.-095505, Aug 27 2001.

[12] K. Dick, T. Dhanasekaran, Z. Y. Zhang, and D. Meisel, “Size-dependent melting of silica-encapsulated gold nanoparticles,” *Journal of the American Chemical Society,* vol. 124, pp. 2312–2317, Mar 13 2002.

[13] F. Ercolessi, W. Andreoni, and E. Tosatti, “Melting of small gold particles–Mechanism and size effects,” *Physical Review Letters,* vol. 66, pp. 911–914, Feb 18 1991.

[14] H. J. Jiang, K. S. Moon, H. Dong, F. Hua, and C. P. Wong, “Size-dependent melting properties of tin nanoparticles,” *Chemical Physics Letters,* vol. 429, pp. 492–496, Oct 5 2006.

[15] Y. Akada, H. Tatsumi, T. Yamaguchi, A. Hirose, T. Morita, and E. Ide, “Interfacial bonding mechanism using silver metallo-organic nanoparticles to bulk metals and observation of sintering behavior,” *Materials Transactions,* vol. 49, pp. 1537–1545, Jul 2008.

[16] J. C. Feng, J. Cao, and Z. R. Li, “Effect of reaction heat on reactive joining of TiAl intermetallics using Ti-Al-C interlayers,” *Scripta Materialia,* vol. 57, pp. 421–424, Sep 2007.

[17] C. Huang, M. F. Becker, J. W. Keto, and D. Kovar, “Annealing of nanostructured silver films produced by supersonic deposition of nanoparticles,” *Journal of Applied Physics,* vol. 102, Sep 1 2007.

[18] S. Jang, J. Joung, and Y. Oh, “Microstructure changes in nanoparticulate gold films under different thermal atmospheres and the effects on bondability,” *Acta Materialia,* vol. 57, pp. 5613–5620, Oct 2009.

[19] R. M. German, *Sintering Theory and Practice*. New York: Wiley, 1996.

[20] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*. Oxford England; New York: Clarendon Press; Oxford University Press, 1987.

[21] C. C. Battaile, “The kinetic Monte Carlo method: Foundation, implementation, and application,” *Computer Methods in Applied Mechanics and Engineering,* vol. 197, pp. 3386–3398, 2008.

[22] S. Plimpton, A. Thompson, and A. Slepoy, “SPPARKS Users Manual,” Sandia National Laboratories, 2008.

[23] D. A. Porter, K. E. Easterling, and M. Y. Sherif, *Phase Transformations in Metals and Alloys*, 3rd ed. Boca Raton, FL: CRC Press, 2009.

[24] C. Garcia Cardona, V. Tikare, and S. Plimpton, “Parallel simulation of 3D sintering,” *Int. J. Computational Materials Science and Surface Engineering,* vol. In Press, 2011.

[25] K. G. F. Janssens, *Computational Materials Engineering: an Introduction to Microstructure Evolution*. Amsterdam ; Boston: Elsevier / Academic Press, 2007.

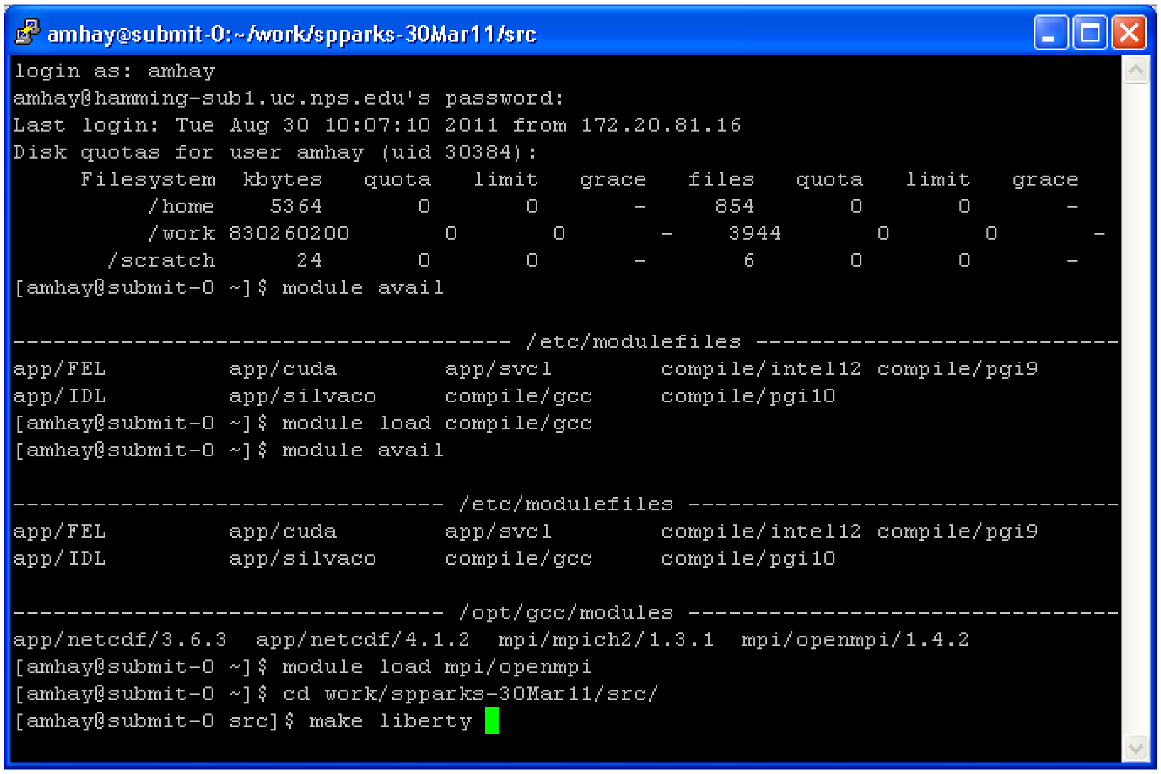
[26] S. Plimpton, C. Battaile, M. Chandross, L. Holm, A. Thompson, V. Tikare, G. Wagner, E. Webb, X. Zhou, C. Garcia Cardona, and A. Slepoy, “Crossing the mesoscale no-man’s land via parallel kinetic monte carlo,” Sandia National Laboratories, SAND2009–6226, 2009.

[27] W. Hoffelner, “Damage assessment in structural metallic materials for advanced nuclear plants,” *Journal of Materials Science,* vol. 45, pp. 2247–2257, May 2010.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. COMPILING SPPARKS ON HAMMING

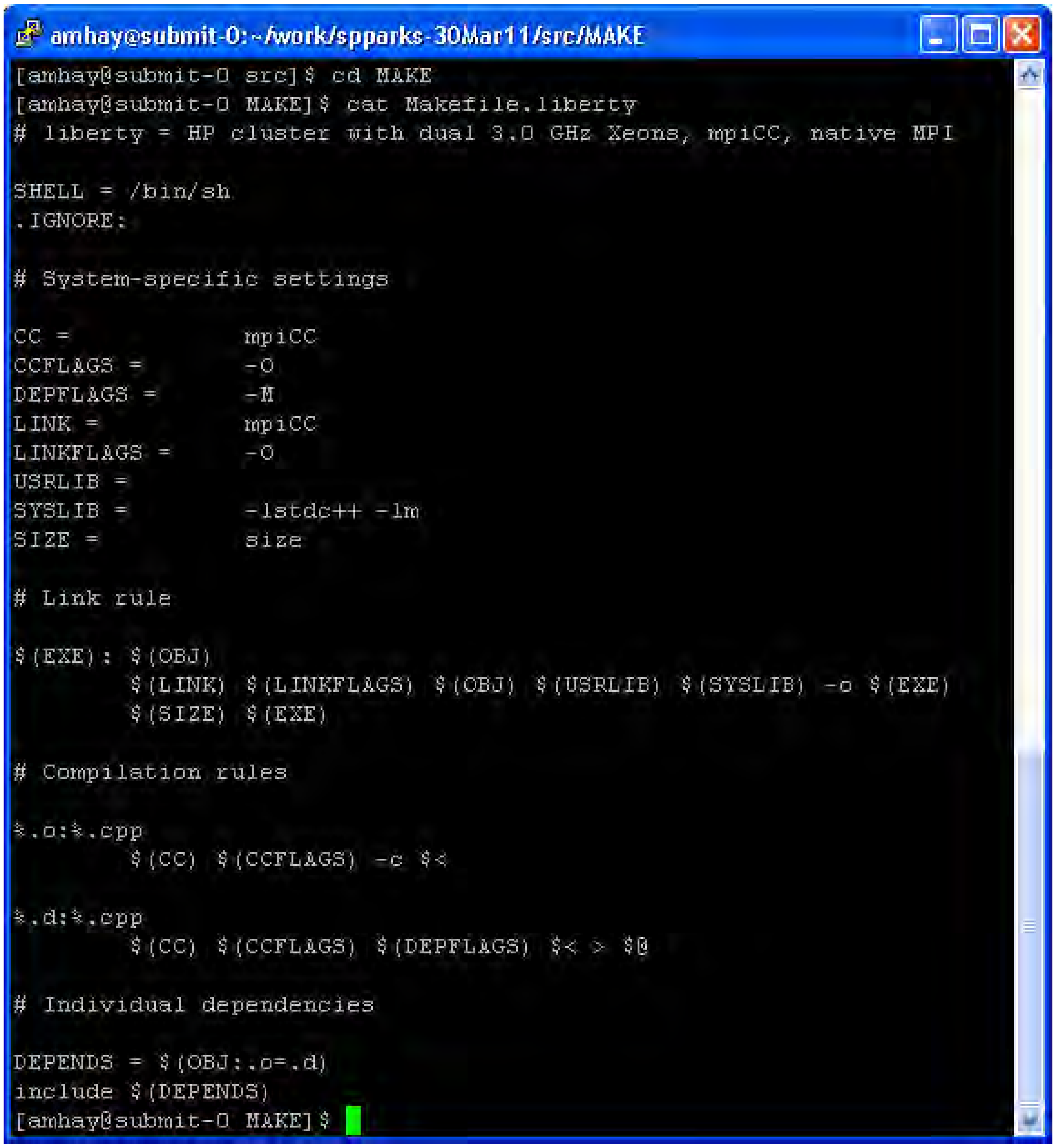
Producing the binary file necessary for running SPPARKS requires the compiling of the code. The specific makefile depends upon the particular machine being used. After logging into the Hamming, the modules compile/gcc and mpi/openmpi are loaded (Figure 42). These files allow SPPARKS to run on multiple processors. The next step is to navigate to the source (src) file under SPPARKS. The makefile 'liberty' is used to generate the executable file. After typing 'make liberty', the code will compile and result in an executable, 'spk_liberty'. The file is then copied into the appropriate example file in order to run simulations. Each time the source code is changed, the code needs to be compiled before running new simulations.



```
login as: amhay
amhay@hamming-sub1.uc.nps.edu's password:
Last login: Tue Aug 30 10:07:10 2011 from 172.20.81.16
Disk quotas for user amhay (uid 30384):
     Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
          /home    5364       0       0       -     854       0       0       -
          /work 830260200       0       0       -    3944       0       0       -
       /scratch      24       0       0       -       6       0       0       -
[amhay@submit-0 ~]$ module avail

------------------------------------- /etc/modulefiles ---------------------------
app/FEL         app/cuda        app/svcl        compile/intel12 compile/pgi9
app/IDL         app/silvaco     compile/gcc     compile/pgi10
[amhay@submit-0 ~]$ module load compile/gcc
[amhay@submit-0 ~]$ module avail

-------------------------------- /etc/modulefiles --------------------------------
app/FEL         app/cuda        app/svcl        compile/intel12 compile/pgi9
app/IDL         app/silvaco     compile/gcc     compile/pgi10

-------------------------------- /opt/gcc/modules --------------------------------
app/netcdf/3.6.3  app/netcdf/4.1.2  mpi/mpich2/1.3.1  mpi/openmpi/1.4.2
[amhay@submit-0 ~]$ module load mpi/openmpi
[amhay@submit-0 ~]$ cd work/spparks-30Mar11/src/
[amhay@submit-0 src]$ make liberty
```

Figure 42. Commands for Compiling SPPARKS on Hamming.

Figure 43 shows the makefile liberty, which is used to compile SPPARKS on Hamming. There are no changes required to the downloaded liberty file to compile the code.


amhay@submit-0:~/work/spparks-30Mar11/src/MAKE

```
[amhay@submit-0 src]$ cd MAKE
[amhay@submit-0 MAKE]$ cat Makefile.liberty
# liberty = HP cluster with dual 3.0 GHz Xeons, mpiCC, native MPI

SHELL = /bin/sh
.IGNORE:

# System-specific settings

CC =            mpiCC
CCFLAGS =       -O
DEPFLAGS =      -M
LINK =          mpiCC
LINKFLAGS =     -O
USRLIB =
SYSLIB =        -lstdc++ -lm
SIZE =          size

# Link rule

$(EXE): $(OBJ)
        $(LINK) $(LINKFLAGS) $(OBJ) $(USRLIB) $(SYSLIB) -o $(EXE)
        $(SIZE) $(EXE)

# Compilation rules

%.o:%.cpp
        $(CC) $(CCFLAGS) -c $<

%.d:%.cpp
        $(CC) $(CCFLAGS) $(DEPFLAGS) $< > $@

# Individual dependencies

DEPENDS = $(OBJ:.o=.d)
include $(DEPENDS)
[amhay@submit-0 MAKE]$
```

Figure 43. SPPARKS Liberty Makefile.

APPENDIX B. SAMPLE SPPARKS INPUT CODE

```
# SPPARKS Sinter Code Filename: input_250_5p30.sinter

seed                   56789

app_style              sinter
dimension              3
lattice                sc/26n 1.0
region                 box block 0 250 0 250 0 250
processors             1 1 8
create_box             box
create_sites           box
set                    i1 unique
# set                  site range 1 125000
# read_sites           start_250_5p0.txt
set                    i1 value 0 fraction 0.3

event_ratios           2.0 1.0 4.0
events_temperatures 1.0 1.0 15.0

time_sinter_start   700

sweep                  random
sector                 yes

diag_style             energy
diag_style             sinter_density
diag_style             cluster

stats                  350
dump                   1 700.0 dump_250_5p30.*.sinter

# diag_style           cluster delt 700.0 stats…
                       no logfreq 1.0 700.0…
                       filename cluster_250_5p30.dat

run                    14000
```

The input script and description of each command are taken from the SPPARKS manual located at the following website: www.cs.sandia.gov/~sjplimp/spparks.html. The first section of the input file contains the parameters for setting up the simulation. The first line of the code is the “seed,” which determines the random number generator for SPPARKS. The random number generator is used to determine the probability of a spin swap if the change in energy is positive and the temperature is greater than zero. The probability of a spin swap is calculated and compared against a number from zero to one based on the random number generator. If the probability is greater than the random number, the new spin is retained. The next line determines the type of application. In this case the "app_style" line is set to sinter, which will run the sinter code for simulating grain growth and sintering. The "dimension" command line can be set to two or three depending on the application. The sintering code only allows for three dimensional problems as will be discussed later. The "lattice" command line determines the arrangements of the sites and number of nearest neighbors. In this example, the "lattice" is square and each site has 26 nearest neighbors with 1.0 lattice sites between neighbors. The "region" describes the size of the model and lists the minimum and maximum x, y, and z coordinates. For this example, there are 250 sites per axis or cube edge for a total of 15,625,000 (250 cubed) sites. The next line determines the number of processors and what coordinates to place them. The processors are placed in the x, y, and z directions. The sample input has 1, 1, 8, which runs eight processors in parallel in the z direction. <u>The x and y processors must be kept at one in order to properly display the microstructure developed separately</u>. The "create_box" command generates a simulation box based on a specified region for on-lattice simulations. The "create_sites" command generates "sites" and assigns spins to each site. The "read_sites" command can be used in place of the "create_box" and "create_sites" command to build the simulation area. The "set" command is used to assign the spin values to the sites. It can be set to contain a specific set of site values such as one to 125,000 or a command such as unique to assign a per-site quantity is set to the site ID, which is effectively a value unique to each site. The second "set" command is used to set a specific site value of zero to 30% of the simulation volume. The value of zero in the sintering code represents a pore and the initial value of 30% represents the green density of the initial microstructure.

The second part of the input file determines the parameters used in running the simulation. The "event_ratios" command determines the ratios for grain growth, pore migration, and pore annihilation, respectively. The "events_temperatures" determines the kT values for grain growth, pore migration, and pore annihilation, respectively. For grain growth, only the grain growth temperature is used and all three temperatures are utilized for sintering. The “time_sinter_start” determines the Monte Carlo step (MCS) when sintering begins. The “sweep” command is set to random which chooses sites at random, one at a time. Other “sweep” commands that can be used are “raster,” which is a sweep of the lattice as a loop over all sites in a predetermined order, and “color,” which partitions the lattice sites into sub-groups or colors which are non-interacting. This sequence means that events on two sites of the same color can be performed simultaneously without conflict. This approach allows the code to run in parallel since events on all sites of the same color can be attempted simultaneously. The “sector” command partitions the portion of the simulation domain owned by each processor into sectors or sub-domains. It is used in parallel simulations and can be used on single processors as well. The final input is the “run” time which determines the MCS when the simulation will terminate.

The third section of the input file defines the types of outputs required from the simulation. The “stats” command defines when the statistics of the simulation will be printed to the screen in MCS. The “diag_style” command determines the diagnostics that will be displayed on each stats line. For this example, the output file (Appendix C) contains the standard outputs along with the total energy of the system, the density of the system, and the cluster data, which includes the number of clusters in the simulation (Nclusters), average number of sites per cluster (N), and average radius of each cluster (R). The “diag_style” command can be used to output a separate file at different intervals. For example, the cluster “diag_style” command can be used to output a file containing the above information and the spin values and quantity of spins for each cluster at a specified interval. The “dump” command outputs the spin value of each lattice site at the specified interval.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SPPARKS SINTERING SIMULATION OUTPUT FILE

The following is the output file for a sintering simulation using SPPARKS. The time is given in MCS and the CPU is given in seconds. The time at the bottom of the output are also given in seconds.

```
SPPARKS (30 Mar 2011)
Created box = (0 0 0) to (250 250 250)
  1 by 1 by 8 processor grid
Creating sites ...
  15625000 sites
  15625000 sites have 26 neighbors
Setting site values ...
  15625000 settings made for i1
Setting site values ...
  4685984 settings made for i1
Setting up run ...
  Time    Naccept      Nreject  Nsweeps  Vmade        CPU       Energy  Density    Nclust       <N>       <R>
     0          0            0        0      0          0  3.51936e+08  0.699861  10171441  1.53616  1.00002
   350   62506641   1499993359      100      0        469  1.81548e+08  0.699792    167166    93.47  3.59823
   700   77670712   3047329288      200      0        934  1.56822e+08  0.699856     99248  157.434  4.31478
     .
     .
     .
 13300  386004808  58988995192     3800   6422  3.48e+03  5.16184e+07  0.896971      8475  1843.66  7.02786
 13650  390382059  60547117941     3900   6284  3.55e+03   5.1276e+07  0.897526      8349  1871.48  7.05801
 14000  394722354  62105277646     4000   6319  3.65e+03  5.09534e+07  0.898083      8225   1899.7  7.07448
Loop time of 3647.88 on 8 procs

Solve time (%) = 1727.05 (47.344)
Update time (%) = 0 (0)
Comm  time (%) = 267.191 (7.32454)
Outpt time (%) = 1090.17 (29.8851)
App   time (%) = 453.484 (12.4314)
Other time (%) = 109.979 (3.01487)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. PROCEDURE FOR CREATING AN INPUT MICROSTRUCTURE

1. Open WinSCP
2. Open PuTTY and log onto Hamming (PuTTY is a free and open source terminal emulator application which can act as a client for the SSH)

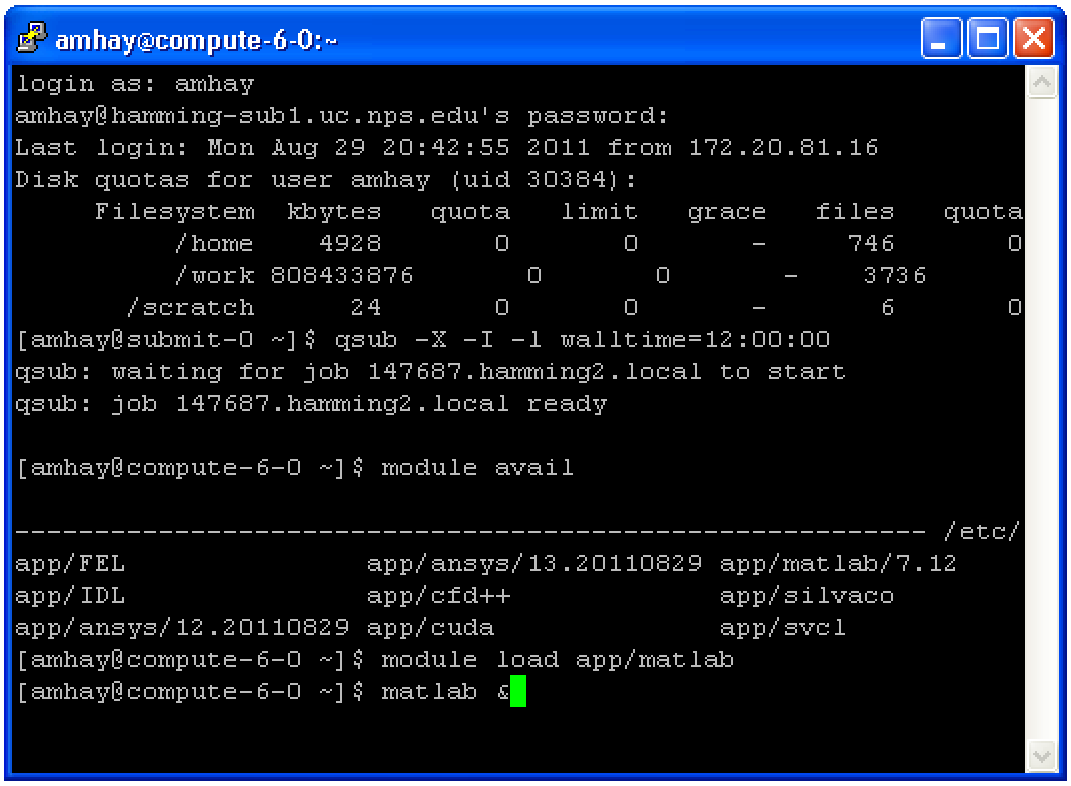
Note: Steps 3–5 are only necessary if the original file needs to be saved.

3. Copy dump file from the Hamming folder to the C: drive
    a. Move “`dump_250_5p30.*.sinter`” into C:SCRATCH
4. Change the name of the file on the C:
    a. “`start_250_5p30.sinter`”
5. Copy the new file back to Hamming folder
6. Open in the file vi editor (`vi start_250_5p30.sinter`)
7. Remove the header from the file and save (`:wq`)
8. Open MATLAB in Hamming (Appendix E)
9. Load dump file into MATLAB
    a. Type “`load start_250_5p30.sinter`” on command line and press enter
10. Remove columns 3–5 of the file
    a. Type “`start_250_5p30(:,3:5)=[];`”
11. Save the MATLAB data table to Hamming as a text file
    a. Type “`dlmwrite(‘start_250_5p30.txt’, start_250_5p30,’delimiter’,’ ‘, ‘precision’,’%9d’)`”
    Note: this step will take approximately 30 minutes for 15,625,000 sites.
12. Open the file in the vi editor (`vi start_250_5p30.txt`)
13. Add the following header:
    a.
```
# Comments

15625000 sites
0 250 xlo xhi
0 250 ylo yhi
0 250 zlo zhi

Values

1 -1
2 -1…
```
14. Save the file (`:wq`)
15. Use this file in the ‘read_sites’ command for the input file

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. LOADING MATLAB ON HAMMING

The four commands in Figure 44 will initiate an interactive node for the user, load and launch MATLAB. The first command requests an interactive node for 12 hours. The default wall time is one hour. After this time, the interactive will close and all applications will terminate. The second command will list the available modules available for load in Hamming. The third command loads MATLAB, and the fourth command launches MATLAB while maintaining the command prompt. MATLAB on Hamming will allow for much larger files to be opened (up to several gigabytes) than MATLAB on the desktop computers. The default for an interactive node is one processor, one gigabyte of memory, and one hour. Figure 44 has a request for 12 hours. Additional processors and memory can be requested by adding "`-l procs=y`" or "`-l nodes=x:ppn=y.`"



```
login as: amhay
amhay@hamming-sub1.uc.nps.edu's password:
Last login: Mon Aug 29 20:42:55 2011 from 172.20.81.16
Disk quotas for user amhay (uid 30384):
     Filesystem  kbytes   quota   limit   grace   files   quota
          /home    4928       0       0       -     746       0
          /work 808433876       0       0       -    3736
       /scratch      24       0       0       -       6       0
[amhay@submit-0 ~]$ qsub -X -I -l walltime=12:00:00
qsub: waiting for job 147687.hamming2.local to start
qsub: job 147687.hamming2.local ready

[amhay@compute-6-0 ~]$ module avail

---------------------------------------------------------- /etc/
app/FEL                app/ansys/13.20110829 app/matlab/7.12
app/IDL                app/cfd++             app/silvaco
app/ansys/12.20110829 app/cuda              app/svcl
[amhay@compute-6-0 ~]$ module load app/matlab
[amhay@compute-6-0 ~]$ matlab &
```

Figure 44. Commands for Loading MATLAB on Hamming.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. SAMPLE HAMMING INPUT CODE

The following is an example of the script used to run SPPARKS on Hamming.

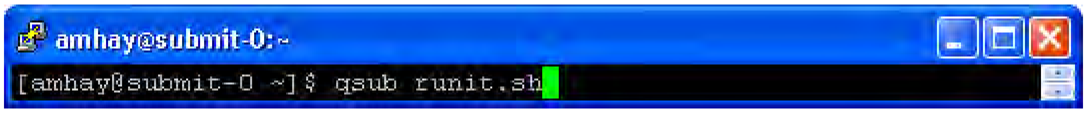
```
#!/bin/bash
#PBS –j oe
#PBS –N Sinter_250_5p30
#PBS –l procs=8
#PBS –l walltime=12:00:00
#PBS –l mem=8gb
#PBS –m be
#
source /etc/profile
module load compile/gcc mpi/openmpi
mpirun .spk_Hamming < ./input_250_5p30.sinter 2>&1…
     >./sinter_250_5p30.txt
```

Table 3 lists the descriptions of each of the options in the above code.

Option	Description
`#PBS –j oe`	Join option that merges the standard error stream with the standard output stream of the job.
`#PBS –N myJob`	Assigns a job name. The default is the name of PBS job script.
`#PBS –l procs=x`	The number of processors. Can also request a number of nodes and processors per node: `#PBS –l nodes=x:ppn=y`.
`#PBS –l walltime=12:00:00`	The maximum wall-clock time during which the job can run.
`#PBS –l mem=8gb`	The maximum memory the job can utilize.
`#PBS –m be`	Sends an e-mail to the user when the job begins and ends.

Table 3. Hamming Input Code Common Commands.

Once the input file is written and the simulation is ready to run, the following is typed on the command line to initiate the simulation (Figure 45).

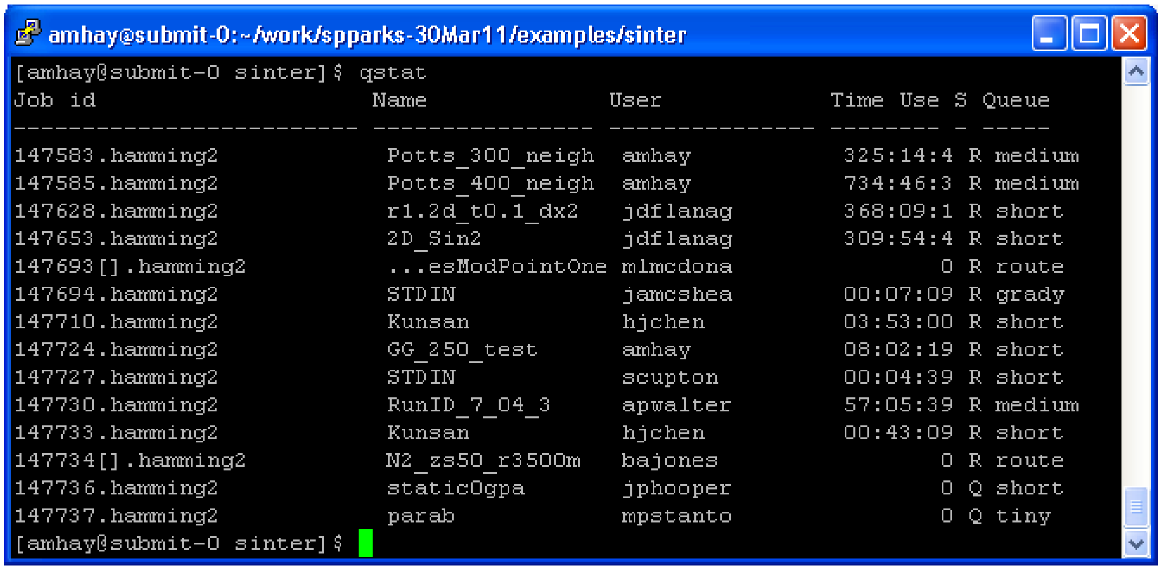


```
[amhay@submit-0 ~]$ qsub runit.sh
```

Figure 45. SPPARKS Simulation Initiation Command in Hamming.

APPENDIX G. COMMON HAMMING COMMANDS

The “qstat” command (Figure 46) is used to list the status of the jobs in Hamming. This command prints a table to the screen containing the job ID, name of the simulation, user, the computer time, status (run=R, Q=queue, C=complete, E=error) and duration of simulation (long > 48 hours, medium = 24–48 hours, short = 2–24 hours, and tiny < 2hours).



amhay@submit-0:~/work/spparks-30Mar11/examples/sinter

```
[amhay@submit-0 sinter]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
147583.hamming2            Potts_300_neigh  amhay           325:14:4 R medium
147585.hamming2            Potts_400_neigh  amhay           734:46:3 R medium
147628.hamming2            r1.2d_t0.1_dx2   jdflanag        368:09:1 R short
147653.hamming2            2D_Sin2          jdflanag        309:54:4 R short
147693[].hamming2          ...esModPointOne mlmcdona               0 R route
147694.hamming2            STDIN            jamcshea        00:07:09 R grady
147710.hamming2            Kunsan           hjchen          03:53:00 R short
147724.hamming2            GG_250_test      amhay           08:02:19 R short
147727.hamming2            STDIN            scupton         00:04:39 R short
147730.hamming2            RunID_7_04_3     apwalter        57:05:39 R medium
147733.hamming2            Kunsan           hjchen          00:43:09 R short
147734[].hamming2          N2_zs50_r3500m   bajones                0 R route
147736.hamming2            static0gpa       jphooper               0 Q short
147737.hamming2            parab            mpstanto               0 Q tiny
[amhay@submit-0 sinter]$
```

Figure 46. Job Stats on Hamming.

Other commands that can be used are:

a. `qstat –a` – This command lists additional information about the jobs such as number of nodes, number of processors, memory allocation, requested run time, and actual run time.

b. `qstat –f 147602` – This command lists all the information for the job including the actual memory used, specific processors being used, compute time, and walltime.

c. `qdel 147602` – This command will delete the job from Hamming.

d. `vi filename` – This command allows the user to edit files in the vi editor of Hamming. It is useful to edit files too large to open with other applications. The following commands are useful in the vi editor:

Command	Description
`:w`	Writes the file to Hamming.
`:q`	Exits the file after saving.
`:q!`	Exits the file without saving.
`:0`	Moves the cursor to the top line of the file.
`:line number`	Moves the curser to the specified line in the file.

Table 4. Common Hamming vi Editor Commands.

APPENDIX H. SIMULATION TIME VERSUS PROCESSORS

Run	Processors	MCS	Sweeps	Clusters	N	R	Total Sites	Memory (GB)	Memory/ Processor
1	6	35,000	10,000	210	74,404.8	32.6371	15,625,000	4.899	0.8165
	8	35,000	10,000	214	73,014.0	32.4512	15,625,000	5.991	0.7489
	10	35,000	10,000	237	65,928.3	33.0723	15,625,000	6.253	0.6253
	12	35,000	10,000	213	73,356.8	32.1083	15,625,000	6.387	0.5323
	14	35,000	10,000	247	63,259.1	31.6558	15,625,000	6.565	0.4689
	16	35,000	10,000	226	69,137.2	33.0697	15,625,000	6.783	0.4239
	18	35,000	10,000	215	72,674.4	33.5101	15,625,000	6.945	0.3858
	20	35,000	10,000	225	69,444.4	32.0559	15,625,000	7.090	0.3545
2	6	35,000	10,000	210	74,404.8	32.6371	15,625,000	5.076	0.8460
	8	35,000	10,000	214	73,014.0	32.4512	15,625,000	5.121	0.6401
	10	35,000	10,000	237	65,928.3	33.0723	15,625,000	5.314	0.5314
	12	35,000	10,000	213	73,356.8	32.1083	15,625,000	6.170	0.5142
	14	35,000	10,000	247	63,259.1	31.6558	15,625,000	6.569	0.4692
	16	35,000	10,000	226	69,137.2	33.0697	15,625,000	6.711	0.4194
	18	35,000	10,000	215	72,674.4	33.5101	15,625,000	5.975	0.3319
	20	35,000	10,000	225	69,444.4	32.0559	15,625,000	6.451	0.3226

Run	Processors	Solve Time (sec)	Update (sec)	Comm (sec)	Output (sec)	App (sec)	Other (sec)	Total (sec)
1	6	5,979.58	0.00	675.29	1,433.87	0.05	0.05	8,088.8
	8	4,406.31	0.00	1,129.31	1,214.79	0.07	0.06	6,750.5
	10	3,519.41	0.00	1,145.44	1,062.48	0.07	0.07	5,727.5
	12	2,827.73	0.00	1,592.03	1,038.17	0.06	0.05	5,458.0
	14	2,445.65	0.00	1,854.11	1,067.88	0.06	0.20	5,367.9
	16	2,150.14	0.00	1,710.50	1,001.05	0.05	0.06	4,861.8
	18	1,881.44	0.00	1,465.07	873.88	0.04	0.06	4,220.5
	20	1,605.63	0.00	728.73	771.62	0.03	0.05	3,106.1
2	6	6,095.18	0.00	826.91	1,460.54	0.06	0.06	8,382.8
	8	4,569.29	0.00	1,004.65	1,193.84	0.07	0.06	6,767.9
	10	3,508.68	0.00	827.64	1,032.24	0.06	0.89	5,369.5
	12	2,975.17	0.00	906.40	935.94	0.07	0.11	4,817.7
	14	2,560.07	0.00	1,067.05	989.16	0.06	0.07	4,616.4
	16	2,357.13	0.00	1,229.96	887.53	0.06	0.08	4,474.8
	18	2,069.11	0.00	1,053.02	819.09	0.05	0.07	3,941.3
	20	1,790.84	0.00	644.33	725.14	0.04	0.06	3,160.4

Table 5. Processors Versus Simulation Time for Sintering.

THIS PAGE INTENTIONALLY LEFT BLANK

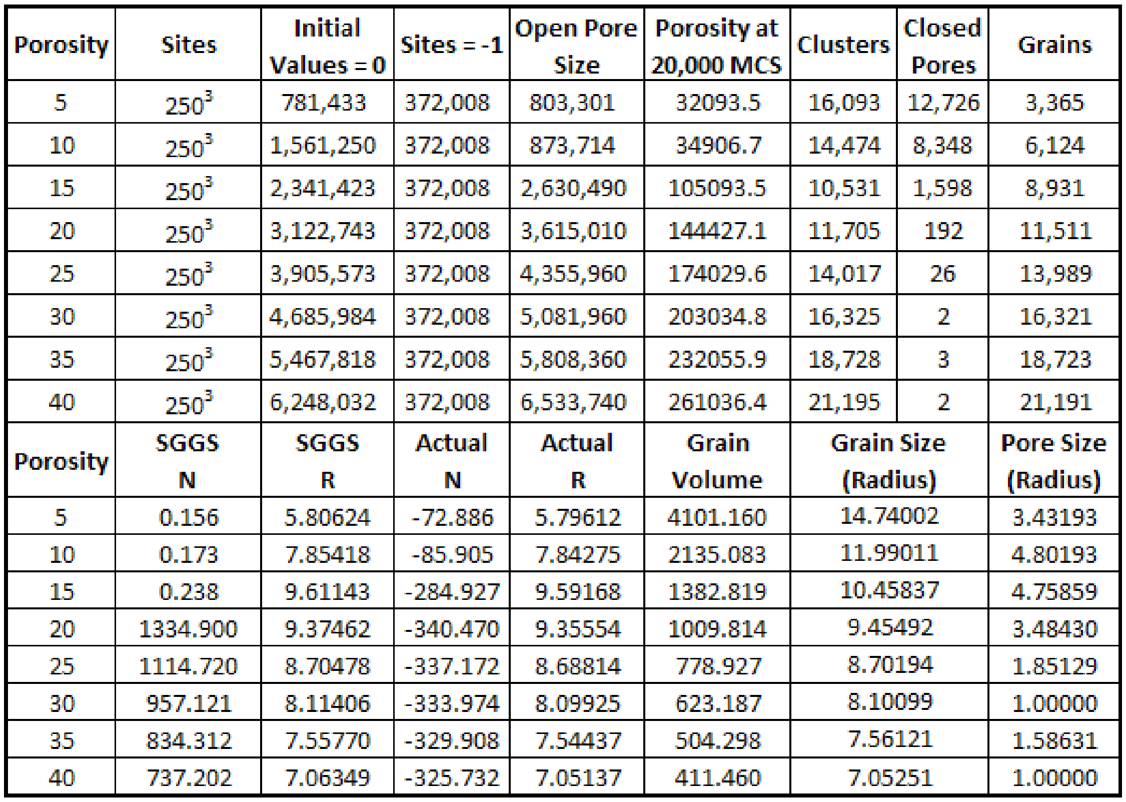
APPENDIX I. SEED DATA

Table 6. Seed Data for SGGS Code.

Sites	Processors	Seed	MCS	Density	Porosity	Clusters	N	R	Time	Solve Time
15,625,000	8	1	0	1.000000	0.000000	14,526,786	1.0756	1.00001	12,201.5	7,834.51
		28290	35,000	1.000000	0.000000	236	66,207.60	32.87130		
			70,000	1.000000	0.000000	96	162,760.00	40.55640		
			0	0.700151	0.299849	10,170,918	1.5362	1.00002	12,116.0	7,041.97
			35,000	0.924437	0.075563	5,456	2,863.82	6.01788		
			70,000	0.929162	0.070838	4,392	3,557.60	5.80182		
		2	0	1.000000	0.000000	14,526,786	1.0756	1.00001	11,968.5	7,817.05
		42397	35,000	1.000000	0.000000	215	72,674.40	33.07570		
			70,000	1.000000	0.000000	101	154,703.00	39.74340		
			0	0.699301	0.300699	10,171,005	1.5623	1.00002	12,235.7	6,996.49
			35,000	0.919045	0.080955	5,372	2,908.60	6.06417		
			70,000	0.924421	0.075579	4,428	3,528.68	5.76362		
		3	0	1.000000	0.000000	14,526,786	1.0756	1.00001	22,746.3	14,732.70
		52314	35,000	1.000000	0.000000	213	73,356.80	34.26120		
			70,000	1.000000	0.000000	76	205,592.00	47.42130		
			0	0.700068	0.299932	10,163,774	1.5373	1.00002	12,852.6	6,985.70
			35,000	0.923149	0.076851	5,445	2,869.61	6.11378		
			70,000	0.928861	0.071139	4,387	3,561.66	5.73347		
		4	0	1.000000	0.000000	14,526,786	1.0756	1.00001	19,998.7	12,629.70
		66051	35,000	1.000000	0.000000	221	70,701.40	33.16030		
			70,000	1.000000	0.000000	89	175,562.00	38.01160		
			0	0.699549	0.300451	10,165,884	1.5370	1.00002	22,745.7	12,782.10
			35,000	0.923643	0.076357	5,291	2,953.13	6.10952		
			70,000	0.928249	0.071751	4,294	3,638.80	5.76655		
		5	0	1.000000	0.000000	14,526,786	1.0756	1.00001	19,606.1	9,282.46
		77958	35,000	1.000000	0.000000	226	69,137.20	32.95850		
			70,000	1.000000	0.000000	83	188,253.00	44.29780		
			0	0.700387	0.299613	10,169,919	1.5364	1.00002	20,501.4	12,312.70
			35,000	0.920374	0.079626	5,322	2,935.93	6.12528		
			70,000	0.925522	0.074478	4,400	3,551.14	5.71982		

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX J. POROSITY DATA



Porosity	Sites	Initial Values = 0	Sites = -1	Open Pore Size	Porosity at 20,000 MCS	Clusters	Closed Pores	Grains
5	250^3	781,433	372,008	803,301	32093.5	16,093	12,726	3,365
10	250^3	1,561,250	372,008	873,714	34906.7	14,474	8,348	6,124
15	250^3	2,341,423	372,008	2,630,490	105093.5	10,531	1,598	8,931
20	250^3	3,122,743	372,008	3,615,010	144427.1	11,705	192	11,511
25	250^3	3,905,573	372,008	4,355,960	174029.6	14,017	26	13,989
30	250^3	4,685,984	372,008	5,081,960	203034.8	16,325	2	16,321
35	250^3	5,467,818	372,008	5,808,360	232055.9	18,728	3	18,723
40	250^3	6,248,032	372,008	6,533,740	261036.4	21,195	2	21,191
Porosity	**SGGS N**	**SGGS R**	**Actual N**	**Actual R**	**Grain Volume**	**Grain Size (Radius)**		**Pore Size (Radius)**
5	0.156	5.80624	-72.886	5.79612	4101.160	14.74002		3.43193
10	0.173	7.85418	-85.905	7.84275	2135.083	11.99011		4.80193
15	0.238	9.61143	-284.927	9.59168	1382.819	10.45837		4.75859
20	1334.900	9.37462	-340.470	9.35554	1009.814	9.45492		3.48430
25	1114.720	8.70478	-337.172	8.68814	778.927	8.70194		1.85129
30	957.121	8.11406	-333.974	8.09925	623.187	8.10099		1.00000
35	834.312	7.55770	-329.908	7.54437	504.298	7.56121		1.58631
40	737.202	7.06349	-325.732	7.05137	411.460	7.05251		1.00000

Table 7. SGGS Porosity Simulations Results.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. MAE Department Chairman,
   Dr. Knox Millsaps
   Naval Postgraduate School
   Monterey, California

4. Engineering and Technology Curricular Office, Code 34
   Naval Postgraduate School
   Monterey, California

5. Professor Luke Brewer
   Naval Postgraduate School
   Monterey, California

6. Professor Young Kwon
   Naval Postgraduate School
   Monterey, California

7. Dr. Veena Tikare
   Sandia National Laboratories
   Albuquerque, New Mexico